Oculus Best Practices

Version 310-30000-02

Copyrights and Trademarks

Contents

Introduction to Best Practices

VR is an immersive medium. It creates the sensation of being entirely transported into a virtual (or real, but digitally reproduced) three-dimensional world, and it can provide a far more visceral experience than screen-based media. These best practices are intended to help developers produce content that provides a safe and enjoyable consumer experience on Oculus hardware. Developers are responsible for ensuring their content conforms to all standards and industry best practices on safety and comfort, and for keeping abreast of all relevant scientific literature on these topics.

Overview

If VR experiences ignore fundamental best practices, they can lead to simulator sickness in some people. Simulator sickness is a combination of symptoms clustered around eyestrain, disorientation, and nausea. Historically, many of these problems have been attributed to sub-optimal VR hardware variables, such as system latency. The Oculus Rift represents a new generation of VR devices, one that resolves many issues of earlier systems. But even with a flawless hardware implementation, improperly designed content can still lead to an uncomfortable experience.

Because VR has been a fairly esoteric and specialized discipline, there are still aspects of it that haven't been studied enough for anybody to make authoritative statements. In these cases, we put forward informed theories and observations and indicate them as such. User testing of your content is absolutely crucial for designing engaging, comfortable experiences; VR as a popular medium is still too young to have established conventions that address every aspect of the experience. Although our researchers have testing underway, there is only so much they can study at a time. We count on you, the community of Oculus Rift developers, to provide feedback and help us mature these evolving VR best practices and principles.

Note: As with any medium, excessive use without breaks is not recommended for developers, end-users, or the device. Please see the latest version of the Health and Safety Warnings at www.oculus.com/warnings.

Rendering

- Use the Oculus VR distortion shaders. Approximating your own distortion solution, even when it "looks about right," is often discomforting for users.
- Get the projection matrix exactly right and use the default Oculus head model. Any deviation from the optical flow that accompanies real world head movement creates oculomotor issues and bodily discomfort.
- Maintain VR immersion from start to finish. For example, don't affix an image in front of the user, such as a full-field splash screen that does not respond to head movements, as this can be disorienting.
- The images presented to each eye should differ only in terms of viewpoint; post-processing effects (e.g., light distortion, bloom) must be applied to both eyes consistently as well as rendered in z-depth correctly to create a properly fused image.
- Consider supersampling and/or anti-aliasing to remedy low apparent resolution, which will appear worst at the center of each eye's screen.

Minimizing Latency

- Your code should run at a frame rate equal to or greater than the Rift display refresh rate, v-synced and unbuffered. Lag and dropped frames produce judder which is discomforting in VR.
- Ideally, target 20ms or less motion-to-photon latency (measurable with the Rift's built-in latency tester). Organize your code to minimize the time from sensor fusion (reading the Rift sensors) to rendering.
- Game loop latency is not a single constant and varies over time. The SDK uses some tricks (e.g., predictive tracking, TimeWarp) to shield the user from the effects of latency, but do everything you can to minimize *variability* in latency across an experience.

- Use the SDK's predictive tracking, making sure you provide an accurate time parameter to the function call. The predictive tracking value varies based on application latency and must be tuned per application.
- Consult the OculusRoomTiny source code as an example for minimizing latency and applying proper rendering techniques in your code.

Optimization

- Decrease eye-render buffer resolution to save video memory and increase frame rate.
- Although dropping display resolution can seem like a good method for improving performance, the resulting benefit comes primarily from its effect on eye-render buffer resolution. Dropping the eye-render buffer resolution while maintaining display resolution can improve performance with less of an effect on visual quality than doing both.

Head-tracking and Viewpoint

- Avoid visuals that upset the user's sense of stability in their environment. Rotating or moving the horizon line or other large components of the user's environment in conflict with the user's real-world self-motion (or lack thereof) can be discomforting.
- The display should respond to the user's movements at all times, without exception. Even in menus, when the game is paused, or during cut scenes, users should be able to look around.
- Use the SDK's position tracking and head model to ensure the virtual cameras rotate and move in a manner consistent with head and body movements; discrepancies are discomforting.

Positional Tracking

- The rendered image must correspond directly with the user's physical movements; do not manipulate the gain of the virtual camera's movements. A single global scale on the entire head model is fine (e.g. to convert feet to meters, or to shrink or grow the player), but do not scale head motion independent of interpupillary distance (IPD).
- With positional tracking, users can now move their viewpoint to look places you might have not expected them to, such as under objects, over ledges, and around corners. Consider your approach to culling and backface rendering, and so on.
- Under certain circumstances, users might be able to use positional tracking to clip through the virtual environment (e.g., put their head through a wall or inside objects). Our observation is that users tend to avoid putting their heads through objects once they realize it is possible, unless they realize an opportunity to exploit game design by doing so. Regardless, developers should plan for how to handle the cameras clipping through geometry. One approach to the problem is to trigger a message telling them they have left the camera's tracking volume (though they technically may still be in the camera frustum).
- Provide the user with warnings as they approach (but well before they reach) the edges of the position camera's tracking volume as well as feedback for how they can re-position themselves to avoid losing tracking.
- We recommend you do not leave the virtual environment displayed on the Rift screen if the user leaves the camera's tracking volume, where positional tracking is disabled. It is far less discomforting to have the scene fade to black or otherwise attenuate the image (such as dropping brightness and/or contrast) before tracking is lost. Be sure to provide the user with feedback that indicates what has happened and how to fix it.
- Augmenting or disabling position tracking is discomforting. Avoid doing so whenever possible, and darken the screen or at least retain orientation tracking using the SDK head model when position tracking is lost.

Accelerations

- Acceleration creates a mismatch among your visual, vestibular, and proprioceptive senses. Minimize the duration and frequency of such conflicts. Make accelerations as short (preferably instantaneous) and infrequent as you can.

- Remember that "acceleration" does not just mean speeding up while going forward; it refers to *any change in the motion of the user*, whether in direction or speed. Slowing down or stopping, turning while moving or standing still, and stepping or getting pushed sideways are all forms of acceleration.
- Have accelerations initiated and controlled by the user whenever possible. Shaking, jerking, or bobbing the camera will be uncomfortable for the player.

Movement Speed

- Viewing the environment from a stationary position is most comfortable in VR; however, when movement through the environment is required, users are most comfortable moving through virtual environments at a constant velocity. Real-world speeds will be comfortable for longer. For reference, humans walk at an average rate of 1.4 m/s.
- Teleporting between two points instead of walking between them is worth experimenting with in some cases, but can also be disorienting. If using teleportation, provide adequate visual cues so users can maintain their bearings, and preserve their original orientation if possible.
- Movement in one direction while looking in another direction can be disorienting. Minimize the necessity for the user to look away from the direction of travel, particularly when moving faster than a walking pace.
- Avoid vertical linear oscillations, which are most discomforting at 0.2 Hz, and off-vertical-axis rotations, which are most discomforting at 0.3 Hz.

Cameras

- Zooming in or out with the camera can induce or exacerbate simulator sickness, particularly if doing so head and camera movements to fall out of 1-to-1 correspondence with each other. We advise against using "zoom" effects until further research and development finds a comfortable and user-friendly implementation.
- For third-person content, be aware that the guidelines for accelerations and movements still apply to the camera regardless of what the avatar is doing. Furthermore, users must always have the freedom to look all around the environment, which can add new requirements to the design of your content.
- Avoid using Euler angles whenever possible; quaternions are preferable. Try looking straight up and straight down to test your camera. It should always be stable and consistent with your head orientation.
- Do not use "head bobbing" camera effects. They create a series of small but uncomfortable accelerations.

Managing and Testing Simulator Sickness

- Test your content with a variety of un-biased users to ensure it is comfortable to a broader audience. As a developer, you are the worst test subject. Repeated exposure to and familiarity with the Rift and your content makes you less susceptible to simulator sickness or content distaste than a new user.
- People's responses and tolerance to sickness vary, and visually induced motion sickness occurs more readily in virtual reality headsets than with computer or TV screens. Your audience will not "muscle through" an overly intense experience, nor should they be expected to do so.
- Consider implementing mechanisms that allow users to adjust the intensity of the visual experience. This will be content-specific, but adjustments might include movement speed, the size of accelerations, or the breadth of the displayed FOV. Any such settings should default to the lowest-intensity experience.
- For all user-adjustable settings related to simulator sickness management, users may want to change them on-the-fly (for example, as they become accustomed to VR or become fatigued). Whenever possible, allow users to change these settings in-game without restarting.
- An independent visual background that matches the player's real-world inertial reference frame (such as a skybox that does not move in response to controller input but can be scanned with head movements) can reduce visual conflict with the vestibular system and increase comfort (see *Tracking* on page 20).
- High spatial frequency imagery (e.g., stripes, fine textures) can enhance the perception of motion in the virtual environment, leading to discomfort. Use—or offer the option of—flatter textures in the environment

(such as solid-colored rather than patterned surfaces) to provide a more comfortable experience to sensitive users.

Degree of Stereoscopic Depth ("3D-ness")

- For individualized realism and a correctly scaled world, use the middle-to-eye separation vectors supplied by the SDK from the user's profile.
- Be aware that depth perception from stereopsis is sensitive up close, but quickly diminishes with distance. Two mountains miles apart in the distance will provide the same sense of depth as two pens inches apart on your desk.
- Although increasing the distance between the virtual cameras can enhance the sense of depth from stereopsis, beware of unintended side effects. First, this will force users to converge their eyes more than usual, which could lead to eye strain if you do not move objects farther away from the cameras accordingly. Second, it can give rise to perceptual anomalies and discomfort if you fail to scale head motion equally with eye separation.

User Interface

- UIs should be a 3D part of the virtual world and sit approximately 2-3 meters away from the viewer—even if it's simply drawn onto a floating flat polygon, cylinder or sphere that floats in front of the user.
- Don't require the user to swivel their eyes in their sockets to see the UI. Ideally, your UI should fit inside the middle 1/3rd of the user's viewing area. Otherwise, they should be able to examine the UI with head movements.
- Use caution for UI elements that move or scale with head movements (e.g., a long menu that scrolls or moves as you move your head to read it). Ensure they respond accurately to the user's movements and are easily readable without creating distracting motion or discomfort.
- Strive to integrate your interface elements as intuitive and immersive parts of the 3D world. For example, ammo count might be visible on the user's weapon rather than in a floating HUD.
- Draw any crosshair, reticle, or cursor at the same depth as the object it is targeting; otherwise, it can appear as a doubled image when it is not at the plane of depth on which the eyes are converged.

Controlling the Avatar

- User input devices can't be seen while wearing the Rift. Allow the use of familiar controllers as the default input method. If a keyboard is absolutely required, keep in mind that users will have to rely on tactile feedback (or trying keys) to find controls.
- Consider using head movement itself as a direct control or as a way of introducing context sensitivity into your control scheme.

Sound

- When designing audio, keep in mind that the output source follows the user's head movements when they wear headphones, but not when they use speakers. Allow users to choose their output device in game settings, and make sure in-game sounds appear to emanate from the correct locations by accounting for head position relative to the output device.
- Presenting NPC (non-player character) speech over a central audio channel or left and right channels equally is a common practice, but can break immersion in VR. Spatializing audio, even roughly, can enhance the user's experience.
- Keep positional tracking in mind with audio design. For example, sounds should get louder as the user leans towards their source, even if the avatar is otherwise stationary.

Content

- For recommendations related to distance, one meter in the real world corresponds roughly to one unit of distance in Unity.
- The optics of the Rift make it most comfortable to view objects that fall within a range of 0.75 to 3.5 meters from the user's eyes. Although your full environment may occupy any range of depths, objects at which users will look for extended periods of time (such as menus and avatars) should fall in that range.
- Converging the eyes on objects closer than the comfortable distance range above can cause the lenses of the eyes to misfocus, making clearly rendered objects appear blurry as well as lead to eyestrain.
- Bright images, particularly in the periphery, can create noticeable display flicker for sensitive users; if possible, use darker colors to prevent discomfort.
- A virtual avatar representing the user's body in VR can have pros and cons. On the one hand, it can increase immersion and help ground the user in the VR experience, when contrasted to representing the player as a disembodied entity. On the other hand, discrepancies between what the user's real-world and virtual bodies are doing can lead to unusual sensations (for example, looking down and seeing a walking avatar body while the user is sitting still in a chair). Consider these factors in designing your content.
- Consider the size and texture of your artwork as you would with any system where visual resolution and texture aliasing is an issue (e.g. avoid very thin objects).
- Unexpected vertical accelerations, like those that accompany traveling over uneven or undulating terrain, can create discomfort. Consider flattening these surfaces or steadying the user's viewpoint when traversing such terrain.
- Be aware that your user has an unprecedented level of immersion, and frightening or shocking content can have a profound effect on users (particularly sensitive ones) in a way past media could not. Make sure players receive warning of such content in advance so they can decide whether or not they wish to experience it.
- Don't rely entirely on the stereoscopic 3D effect to provide depth to your content. Lighting, texture, parallax (the way objects appear to move in relation to each other when the user moves), and other visual features are equally (if not more) important to conveying depth and space to the user. These depth cues should be consistent with the direction and magnitude of the stereoscopic effect.
- Design environments and interactions to minimize the need for strafing, back-stepping, or spinning, which can be uncomfortable in VR.
- People will typically move their heads/bodies if they have to shift their gaze and hold it on a point farther than 15-20° of visual angle away from where they are currently looking. Avoid forcing the user to make such large shifts to prevent muscle fatigue and discomfort.
- Don't forget that the user is likely to look in any direction at any time; make sure they will not see anything that breaks their sense of immersion (such as technical cheats in rendering the environment).

Avatar Appearance

- When creating an experience, you might choose to have the player experience it as a ghost (no physical presence) or in a body that is very different from his or her own. For example, you might have a player interact with your experience as a historical figure, a fictional character, a cartoon, a dragon, a giant, an orc, an amoeba, or any other of a multitude of possibilities. Any such avatars should not create issues for users as long as you adhere to best practices guidelines for comfort and provide users with intuitive controls.
- When the avatar is meant to represent the players themselves inside the virtual environment, it can detract from immersion if the player looks down and sees a body or hands that are very different than his or her own. For example, a woman's sense of immersion might be broken if she looks down and sees a man's hands or body. Allowing players to customize their hands and bodies can dramatically improve immersion. If this adds too much cost or complexity to your project, you can still take measures to minimize contradictions between VR and reality. For example, avoid overtly masculine or feminine bodily features in visible parts of the avatar. Gloves and unisex clothing that fit in the theme of your content can also serve to maintain ambiguity in aspects of the avatar's identity, such as gender, body type, and skin color.

Health and Safety

Carefully read and implement the warnings that accompany the Rift (see *www.oculus.com/warnings*) to ensure the health and safety for you, anyone testing your content, and your users.

Image Safety and Photosensitive Seizures

Certain types of images are believed to be capable of triggering photosensitive seizures in a small portion of the population.[1] The International Standards Organization is in the process of developing a standard for image content to reduce the risk of photosensitive seizures. You are responsible for staying abreast of the standards and literature on photosensitive seizures and image safety and designing your content to conform to the standards and recommended best practices on these subjects.

[1] International Standard ISO/DIS 9241-391.2, Ergonomics of Human System Interaction – Part 391: Requirements, analysis and compliance test methods for the reduction of photosensitive seizures (approved and published pending periodical review).

Binocular Vision, Stereoscopic Imaging and Depth Cues

- *The brain uses differences between your eyes' viewpoints to perceive depth.*
- *Don't neglect monocular depth cues, such as texture and lighting.*
- *The most comfortable range of depths for a user to look at in the Rift is between 0.75 and 3.5 meters (1 unit in Unity = 1 meter).*
- *Set the distance between the virtual cameras to the distance between the user's pupils from the OVR config tool.*
- *Make sure the images in each eye correspond and fuse properly Effects that appear in only one eye or differ significantly between the eyes look abnormal.*

Basics

Binocular vision describes the way in which we see two views of the world simultaneously—the view from each eye is slightly different and our brain combines them into a single three-dimensional *stereoscopic image*, an experience known as *stereopsis*. The difference between what we see from our left eye and what we see from our right eye generates *binocular disparity*. Stereopsis occurs whether we are seeing different viewpoints of the physical world from each of our eyes or two flat pictures with appropriate differences (disparity) between them.

The Oculus Rift presents two images, one to each eye, generated by two virtual cameras separated by a short distance. Defining some terminology is in order. The distance between our two eyes is called the *interpupillary distance* (IPD), and we refer to the distance between the two rendering cameras that capture the virtual environment as the inter-camera distance (ICD). Although the IPD can vary from about 52mm to 78mm, average IPD (based on data from a survey of approximately 4,000 U.S. Army soldiers) is about 63.5 mm—the same as the Rift's average *interaxial distance (IAD)*, which is the distance between the centers of the Rift's lenses (as of this revision of this guide).

Monocular depth cues

Stereopsis is just one of many depth cues our brains process. Most of the other depth cues are *monocular*; that is, they convey depth even when they are viewed by only one eye or appear in a flat image viewed by both eyes. For VR, motion parallax due to head movement does not require stereopsis to see, but is extremely important for conveying depth and providing a comfortable experience to the user.

Other important depth cues include: *curvilinear perspective* (straight lines converge as they extend into the distance), *relative scale* (objects get smaller when they are farther away), *occlusion* (closer objects block our view of more distant objects), *aerial perspective* (distant objects appear fainter than close objects due to the refractive properties of the atmosphere), *texture gradients* (repeating patterns get more densely packed as they recede) and *lighting* (highlights and shadows help us perceive the shape and position of objects). Current-generation computer-generated content already leverages a lot of these depth cues, but we mention them because it can be easy to neglect their importance in light of the novelty of stereoscopic 3D.

Comfortable Viewing Distances Inside the Rift

Two issues are of primary importance to understanding eye comfort when the eyes are fixating on (i.e., looking at) an object: *accommodative demand* and *vergence demand*. Accommodative demand refers to how your eyes have to adjust the shape of their lenses to bring a depth plane into focus (a process known as *accommodation*). Vergence demand refers to the degree to which the eyes have to rotate inwards so their lines of sight intersect at a particular depth plane. In the real world, these two are strongly correlated with one another; so much so that we have what is known as the *accommodation-convergence reflex*: the degree of convergence of your eyes influences the accommodation of your lenses, and vice-versa.

The Rift, like any other stereoscopic 3D technology (e.g., 3D movies), creates an unusual situation that decouples accommodative and vergence demands—accommodative demand is *fixed*, but vergence demand can *change*. This is because the actual images for creating stereoscopic 3D are always presented on a screen that remains at the same distance optically, but the different images presented to each eye still require the eyes to rotate so their lines of sight converge on objects at a variety of different depth planes.

Research has looked into the degree to which the accommodative and vergence demands can differ from each other before the situation becomes uncomfortable to the viewer.[1] The current optics of the Rift are equivalent to looking at a screen approximately 1.3 meters away. (Manufacturing tolerances and the power of the Rift's lenses means this number is only a rough approximation.) In order to prevent eyestrain, objects that you know the user will be fixating their eyes on for an extended period of time (e.g., a menu, an object of interest in the environment) should be rendered between approximately 0.75 and 3.5 meters away.

Obviously, a complete virtual environment requires rendering some objects outside this optimally comfortable range. As long as users are not required to fixate on those objects for extended periods, they are of little concern. When programming in Unity, 1 unit will correspond to approximately 1 meter in the real world, so objects of focus should be placed 0.75 to 3.5 distance units away.

Our ongoing research and development might allow future incarnations of the Rift to improve their optics to widen the range of comfortable viewing distances. No matter how this range changes, however, 2.5 meters should be a comfortable distance, making it a safe, future-proof distance for fixed items on which users will have to focus for an extended time, like menus or GUIs.

Anecdotally, some Rift users have remarked on the unusualness of seeing all objects in the world in focus when the lenses of their eyes are accommodated to the depth plane of the virtual screen. This can potentially lead to frustration or eye strain in a minority of users, as their eyes may have difficulty focusing appropriately.

Some developers have found that depth-of-field effects can be both immersive and comfortable for situations in which you know where the user is looking. For example, you might artificially blur the background behind a menu the user brings up, or blur objects that fall outside the depth plane of an object being held up for examination. This not only simulates the natural functioning of your vision in the real world, it can prevent distracting the eyes with salient objects outside the user's focus.

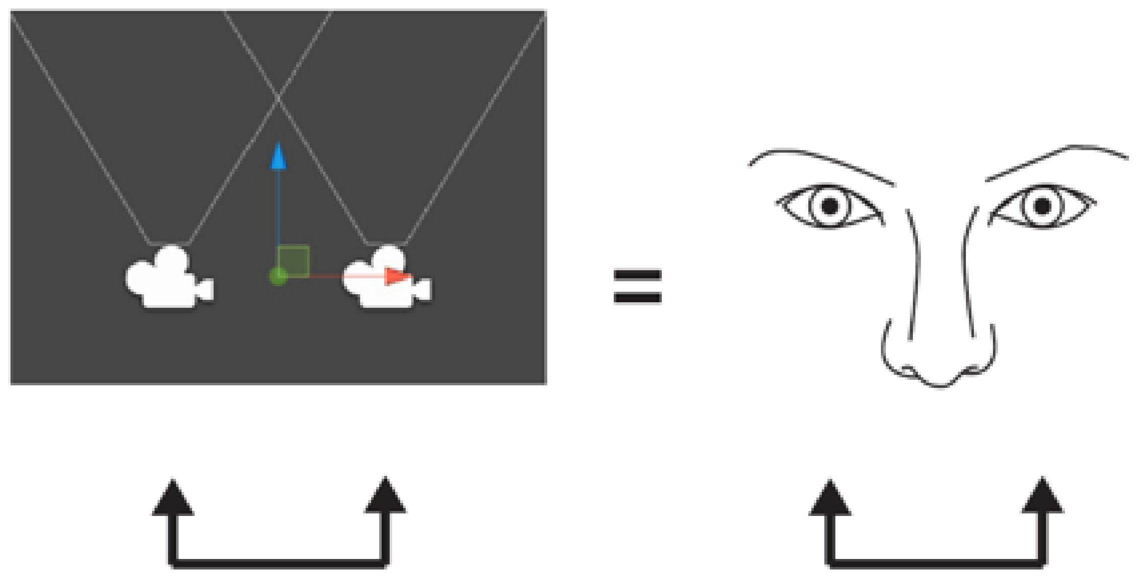
We have no control over a user who chooses to behave in an unreasonable, abnormal, or unforeseeable manner. Someone in VR might choose to stand with their eyes inches away from an object and stare at it all day. Although we know this can lead to eye strain, drastic measures to prevent this anomalous case, such as setting collision detection to prevent users from walking that close to objects, would only hurt overall user experience. Your responsibility as a developer, however, is to avoid requiring the user to put themselves into circumstances we know are sub-optimal.

Effects of Inter-Camera Distance

Changing *inter-camera distance*, the distance between the two rendering cameras, can impact users in important ways. If the inter-camera distance is increased, it creates an experience known as *hyperstereo* in which depth is exaggerated; if it is decreased, depth will flatten, a state known as *hypostereo*. Changing inter-camera distance has two further effects on the user. First, it changes the degree to which the eyes must converge to look at a given object. As you increase inter-camera distance, users have to converge their eyes more to look at the same object, and that can lead to eyestrain. Second, it can alter the user's sense of their own size inside the virtual environment. The latter is discussed further in *Content Creation* under *User and Environment Scale*.

Set the inter-camera distance to the user's actual IPD to achieve veridical scale and depth in the virtual environment. If applying a scaling effect, make sure it is applied to the entire head model to accurately reflect the user's real-world perceptual experience during head movements, as well as any of our guidelines related to distance.

The inter-camera distance (ICD) between the left and right scene cameras (left) must be proportional to the user's inter-pupillary distance (IPD; right). Any scaling factor applied to ICD must be applied to the entire head model and distance-related guidelines provided throughout this guide.



Potential Issues with Fusing Two Images

We often face situations in the real world where each eye gets a very different viewpoint, and we generally have little problem with it. Peeking around a corner with one eye works in VR just as well as it does in real life. In fact, the eyes' different viewpoints can be beneficial: say you're a special agent (in real life or VR) trying to stay hidden in some tall grass. Your eyes' different viewpoints allow you to look "through" the grass to monitor your surroundings as if the grass weren't even there in front of you. Doing the same in a video game on a 2D screen, however, leaves the world behind each blade of grass obscured from view.

Still, VR (like any other stereoscopic imagery) can give rise to some potentially unusual situations that can be annoying to the user. For instance, rendering effects (such as light distortion, particle effects, or light bloom) should appear in both eyes and with correct disparity. Failing to do so can give the effects the appearance of flickering/shimmering (when something appears only in one eye) or floating at the wrong depth (if disparity is off, or if the post processing effect is not rendered to contextual depth of the object it should be effecting—for example, a specular shading pass). It is important to ensure that the images between the two eyes do not differ aside from the slightly different viewing positions inherent to binocular disparity.

Although less likely to be a problem in a complex 3D environment, it can be important to ensure the user's eyes receive enough information for the brain to know how to fuse and interpret the image properly. The lines and edges that make up a 3D scene are generally sufficient; however, be wary of wide swaths of repeating patterns, which could cause people to fuse the eyes' images differently than intended. Be aware also that optical illusions of depth (such as the "hollow mask illusion," where concave surfaces appear convex) can sometimes lead to misperceptions, particularly in situations where monocular depth cues are sparse.

[1] Shibata, T., Kim, J., Hoffman, D.M., Banks, M.S. (2011). The zone of comfort: Predicting visual discomfort with stereo displays. *Journal of Vision, 11(8),* 1-29.

Field of View and Scale

The FOV of the virtual cameras must match the visible display area. In general, Oculus recommends not changing the default FOV.

Field of view can refer to different things that we will first disambiguate. If we use the term *display field of view* (dFOV), we are referring to the part of the user's physical visual field occupied by VR content. It is a physical characteristic of the hardware and optics. The other type of FOV is *camera field of view* (cFOV), which refers to the range of the virtual world that is seen by the rendering cameras at any given moment. All FOVs are defined by an angular measurement of vertical, horizontal, and/or diagonal dimensions.

In ordinary screen-based computer graphics, you usually have the freedom to set the camera's cFOV to anything you want: from fisheye (wide angle) all the way to telephoto (narrow angle). Although people can experience some visually-induced motion sickness from a game on a screen,[1] this typically has little effect on many users because the image is limited to an object inside the observer's total view of the environment. Computer users' peripheral vision can see the room that their display sits in, and the monitor typically does not respond to their head movements. While the image may be immersive, the brain is not usually fooled into thinking it is actually real, and differences between cFOV and dFOV do not cause problems for the majority of people.

In virtual reality, there is no view of the external room, and the virtual world fills much of your peripheral vision. It is therefore very important that the cFOV and the dFOV match exactly. The ratio between these two values is referred to as the *scale*, and in virtual reality the scale should always be exactly 1.0.

In the Rift, the maximum dFOV is determined by the screen, the lenses, and how close the user puts the lenses to their eyes (in general, the closer the eyes are to the lens, the wider the dFOV). The configuration utility measures the maximum dFOV that users can see, and this information is stored inside their profile. The SDK will recommend a cFOV that matches the dFOV based on this information.

Note: Because some people have one eye closer to the screen than the other, each eye can have a different dFOV. This is normal.

Deviations between dFOV and cFOV have been found to be discomforting[2] (though some research on this topic has been mixed[3]). If scale deviates from 1.0, the distortion correction values will cause the rendered scene to warp. Manipulating the camera FOV can also induce simulator sickness and can even lead to a maladaptation in the vestibular-ocular reflex, which allows the eyes to maintain stable fixation on an object during head movements. The maladaptation can make the user feel uncomfortable during the VR experience, as well as impact visual-motor functioning after removing the Rift.

The SDK will allow manipulation of the cFOV and dFOV without changing the scale, and it does so by adding black borders around the visible image. Using a smaller visible image can help increase rendering performance or serve special effects. Just be aware that if you select a 40° visible image, most of the screen will be black—that is entirely intentional and not a bug. Also note that reducing the size of the visible image will require users to look around using head movements more than they would if the visible image were larger; this can lead to muscle fatigue and simulator sickness.

Some games require a "zoom" mode for binoculars or sniper scopes. This is extremely tricky in VR, and must be done with a lot of caution, as a naive implementation of zoom causes disparity between head motion and apparent optical motion of the world, and can cause a lot of discomfort. Look for future blog posts and demos on this.

[1] Stoffregen, T.A., Faugloire, E., Yoshida, K., Flanagan, M.B., & Merhi, O. (2008). Motion sickness and postural sway in console video games. *Human Factors, 50,* 322-331.

[2] Draper, M.H., Viire, E.S., Furness, T.A., Gawron, V.J. (2001). Effects of image scale and system time delay on simulator sickness with head-coupled virtual environments. *Human Factors, 43(1)*, 129-146.

[3] Moss, J. D., & Muth, E. R. (2011). Characteristics of Head-Mounted Displays and Their Effects on Simulator Sickness. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 53(3)*, 308–319.

Rendering Techniques

Be mindful of the Rift screen's resolution, particularly with fine detail. Make sure text is large and clear enough to read and avoid thin objects and ornate textures in places where users will focus their attention.

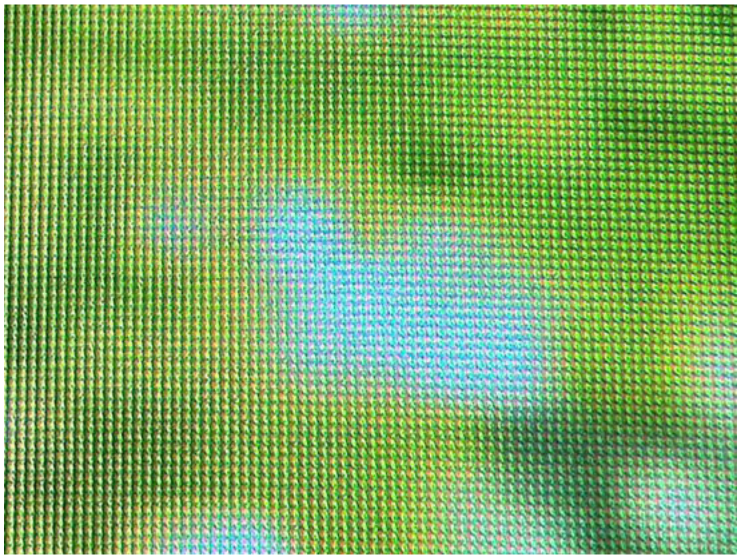
Display Resolution

The current Rift has a 2160 x 1200 low-persistence OLED display with a 90-hz refresh rate. This represents a leap forward from the original DK1 in many respects, which featured a 1280 x 720, full-persistence 60-hz LCD display. The higher resolution means images are clearer and sharper, while the low persistence and high refresh rate eliminate much of the *motion blur* (i.e., blurring when moving your head) found in DK1.

The DK1 panel, which uses a grid pixel structure, gives rise to a "screen door effect" (named for its resemblance to looking through a screen door) due to the space between pixels. The Rift, on the other hand, has a pentile structure that produces more of a honeycomb-shaped effect. Red colors tend to magnify the effect due to the unique geometry of the display's sub-pixel separation.

Combined with the effects of lens distortion, some detailed images (such as text or detailed textures) may look different inside the Rift than on your computer monitor. Be sure to view your artwork and assets inside the Rift during the development process and make any adjustments necessary to ensure their visual quality.

Figure 1: "Screen Door" Effect



Understanding and Avoiding Display Flicker

Display flicker is generally perceived as a rapid "pulsing" of lightness and darkness on all or parts of a screen. Some people are extremely sensitive to flicker and experience eyestrain, fatigue, or headaches as a result. Others will never even notice it or have any adverse symptoms. Still, there are certain factors that can increase or decrease the likelihood any given person will perceive display flicker.

The degree to which a user will perceive flicker is a function of several factors, including: the rate at which the display is cycling between "on" and "off" modes, the amount of light emitted during the "on" phase, how much of which parts of the retina are being stimulated, and even the time of day and fatigue level of the individual.

Two pieces of information are important to developers. First, people are more sensitive to flicker in the periphery than in the center of vision. Second, brighter screen images produce more flicker. Bright imagery, particularly in the periphery (e.g., standing in a bright, white room) can potentially create noticeable display flicker. Try to use darker colors whenever possible, particularly for areas outside the center of the player's viewpoint.

The higher the refresh rate, the less perceptible flicker is. This is one of the reasons it is so critical to run at 75fps v-synced, unbuffered. As VR hardware matures over time, refresh rate and frame rate will very likely exceed 75fps.

Rendering resolution

The Rift has a display resolution of 2160 x 1200, but the distortion of the lenses means the rendered image on the screen must be transformed to appear normal to the viewer. In order to provide adequate pixel density for the transformation, each eye requires a rendered image that is actually larger than the resolution of its half of the display.

Such large render targets can be a performance problem for some graphics cards, and dropping frame rate produces a poor VR experience. Dropping display resolution has little effect, and can introduce visual artifacts. Dropping the resolution of the eye buffers, however, can improve performance while maintaining perceived visual quality.

This process is covered in more detail in the SDK.

Dynamically-rendered impostors/billboards

Depth perception becomes less sensitive at greater distances from the eyes. Up close, stereopsis might allow you to tell which of two objects on your desk is closer on the scale of millimeters. This becomes more difficult further out. If you look at two trees on the opposite side of a park, they might have to be meters apart before you can confidently tell which is closer or farther away. At even larger scales, you might have trouble telling which of two mountains in a mountain range is closer to you until the difference reaches kilometers.

You can use this relative insensitivity to depth perception in the distance to free up computational power by using "imposter" or "billboard" textures in place of fully 3D scenery. For instance, rather than rendering a distant hill in 3D, you might simply render a flat image of the hill onto a single polygon that appears in the left and right eye images. This image appears to the eyes in VR the same as in traditional 3D games.

Note: The effectiveness of these imposters will vary depending on the size of the objects involved, the depth cues inside of and around those objects, and the context in which they appear.[1] You will need to engage in individual testing with your assets to ensure the imposters look and feel right. Be wary that the impostors are sufficiently distant from the camera to blend in inconspicuously, and that interfaces between real and impostor scene elements do not break immersion.

Normal Mapping vs. Parallax Mapping

The technique known as "normal mapping" provides realistic lighting cues to convey depth and texture without adding to the vertex detail of a given 3D model. Although widely used in modern games, it is much less compelling when viewed in stereoscopic 3D. Because normal mapping does not account for binocular disparity or motion parallax, it produces an image akin to a flat texture painted onto the object model.

"Parallax mapping" builds on the idea of normal mapping, but accounts for depth cues normal mapping does not. Parallax mapping shifts the texture coordinates of the sampled surface texture by using an additional height map provided by the content creator. The texture coordinate shift is applied using the per-pixel or per-vertex view direction calculated at the shader level. Parallax mapping is best utilized on surfaces with fine detail that would not affect the collision surface, such as brick walls or cobblestone pathways.

[1] Allison, R. S., Gillam, B. J., & Vecellio, E. (2009). Binocular depth discrimination and estimation beyond interaction space. *Journal of Vision, 9,* 1–14.

Motion

Overview

- *The most comfortable VR experiences involve no self-motion for the user besides head and body movements to look around the environment.*
- *When self-motion is required, slower movement speeds (walking/jogging pace) are most comfortable for new users.*
- *Keep any form of acceleration as short and infrequent as possible.*
- *User and camera movements should never be decoupled.*
- *Don't use head bobbing in first-person games.*
- *Experiences designed to minimize the need for moving backwards or sideways are most comfortable.*
- *Beware situations that visually induce strong feelings of motion, such as stairs or repeating patterns that move across large sections of the screen.*

Speed of Movement and Acceleration

"Movement" here refers specifically to any motion through the virtual environment that is *not* the result of mapping the user's real world movements into VR. Movement and acceleration most commonly come from the user's avatar moving through the virtual environment (by locomotion or riding a vehicle) while the user's real-world body is stationary. These situations can be discomforting because the user's vision tells them they are moving through space, but their bodily senses (vestibular sense and proprioception) say the opposite. This illusory perception of self-motion from vision alone has been termed *vection*, and is a major underlying cause of simulator sickness.[1]

Speed of movement through a virtual environment has been found to be proportional to the speed of *onset* for simulator sickness, but not necessarily the subsequent intensity or rate of increase.[2] Whenever possible, we recommend implementing movement speeds near typical human locomotion speeds (about 1.4 m/s walking, 3 m/s for a continuous jogging pace) as a user-configurable—if not default—option.

For VR content, the visual perception of acceleration is a primary impetus for discomfort. This is because the human vestibular system responds to acceleration but not constant velocity. Perceiving acceleration visually without actually applying acceleration to your head or body can lead to discomfort. (See our section on simulator sickness for a more detailed discussion.)

Keep in mind that "acceleration" can refer to any change over time in the velocity of the user in the virtual world in any direction. Although we normally think of acceleration as "increasing the speed of forward movement," acceleration can also refer to decreasing the speed of movement or stopping; rotating, turning, or tilting while stationary or moving; and moving (or ceasing to move) sideways or vertically. It is any change in direction or speed.

Instantaneous accelerations are more comfortable than gradual accelerations. Because any period of acceleration constitutes a period of conflict between the senses, discomfort will increase as a function of the frequency, size, and duration of acceleration. We generally recommend you minimize the duration and frequency of accelerations as much as possible.

Degree of Control

Similar to how drivers are much less likely to experience motion sickness in a car than their passengers, giving users control over the motion they see can prevent simulator sickness. Let users move themselves around instead of taking them for a ride, and avoid jerking the camera around, such as when the user is hit or shot. This can be very effective on a monitor, but can cause simulator sickness. Similarly, do not freeze the display so that it does not respond to the user's head movements, as this can create discomforting misperceptions of illusory motion. In general, avoid decoupling the user's and camera's movements for any reason.

Research suggests that providing users with an avatar that anticipates and foreshadows the visual motion they are about to experience allows them to prepare for it in a way that reduces discomfort. This can be a serendipitous benefit in 3rd-person games. If the avatar's actions (e.g., a car begins turning, a character starts running in a certain direction) reliably predict what the camera is about to do, this may prepare the user for the impending movement through the virtual environment and make for a more comfortable experience.

Head Bobbing

Some first-person games apply a mild up-and-down movement to the camera to simulate the effects of walking. This can be effective to portray humanoid movement on a computer or television screen, but can be a problem for many people in immersive head-mounted VR. Every bob up and down is another bit of acceleration applied to the user's view, which—as we already said above—can lead to discomfort. Do not use any head-bob or changes in orientation or position of the camera that were not initiated by the real-world motion of the user's head.

Forward and lateral movement

In the real world, we most often stand still or move forward. We rarely back up, and we almost never strafe (move side to side). Therefore, when movement is a must, forward user movement is most comfortable. Left or right lateral movement is more problematic because we don't normally walk sideways and it presents an unusual optic flow pattern to the user.

In general, you should respect the dynamics of human motion. There are limits to how people can move in the real world, and you should take this into account in your designs.

Moving up or down stairs (or steep slopes) can be discomforting for people. In addition to the unusual sensation of vertical acceleration, the pronounced horizontal edges of the steps fill the visual field of the display while all moving in the same direction. This creates an intense visual that drives a strong sense of vection. Users do not typically see imagery like this except for rare situations like looking directly at a textured wall or floor while walking alongside it. We recommend that developers use slopes and stairs sparingly. This recommendation applies to other images that strongly induce vection, as well, such as moving up an elevator shaft where stripes (of light or texture) are streaming downwards around the user.

Developers are advised to consider how these guidelines can impact one another in implementation. For example, eliminating lateral and backwards movement from your control scheme might seem like a reasonable idea in theory, but could cause users to engage in relatively more motions (i.e., turning, moving forward, and turning again) to accomplish the same changes in position. This results in more visual self-motion—and consequently more vection—than users would have seen if they simply stepped backwards or to the side. Environments and experiences should be designed to minimize the impact of these issues.

Consider also simplifying complex actions to minimize the amount of vection the user will experience, such as automating or streamlining a complex maneuver for navigating obstacles. One study had players navigate a virtual obstacle course with one of two control schemes: one that gave them control over 3 degrees of freedom in motion, or another that gave them control over 6. Although the 3-degrees-of-freedom control scheme initially seems to give the user less control (and therefore lead to more simulator sickness), it actually led to *less* simulator sickness because it saved them from having to experience extraneous visual motion.[2]

This is one of those cases where a sweeping recommendation cannot be made across different types of content and situations. Careful consideration, user testing, and iterative design are critical to optimizing user experience and comfort.

[1]Hettinger, L.J., Berbaum, K.S., Kennedy, R.S., Dunlap, W.P., & Nolan, M.D. (1990). Vection and simulator sickness. Military Psychology, 2(3), 171-181.

[2]Stanney, K.M. & Hash, P. (1998). Locus of user-initiated control in virtual environments: Influences on cybersickness. Presence, 7(5), 447-459.

Tracking

The FOV of the virtual cameras must match the visible display area. In general, Oculus recommends not changing the default FOV.

Overview

- *The Rift sensors collect information about user yaw, pitch, and roll.*
- *6DoF position tracking to the Rift.*
    - *Allow users to set the origin point based on a comfortable position for them with guidance for initially positioning themselves.*
    - *Do not disable or modify position tracking, especially while the user is moving in the real world.*
    - *Warn the user if they are about to leave the camera tracking volume; fade the screen to black before tracking is lost.*
- *Implement the "head model" code available in our SDK demos whenever position tracking is unavailable.*
- *Optimize your entire engine pipeline to minimize lag and latency.*
- *Implement Oculus VR's predictive tracking code (available in the SDK demos) to further reduce latency.*
- *If latency is truly unavoidable, variable lags are worse than a consistent one.*

Orientation Tracking

The Oculus Rift headset contains a gyroscope, accelerometer, and magnetometer. We combine the information from these sensors through a process known as *sensor fusion* to determine the orientation of the user's head in the real world, and to synchronize the user's virtual perspective in real-time. These sensors provide data to accurately track and portray yaw, pitch, and roll movements.

We have found a very simple model of the user's head and neck to be useful in accurately translating sensor information from head movements into camera movements. We refer to this in short as *the head model*, and it reflects the fact that movement of the head in any of the three directions actually pivots around a point roughly at the base of your neck—near your voice-box. This means that rotation of the head also produces a translation at your eyes, creating motion parallax, a powerful cue for both depth perception and comfort.

Position Tracking

The Rift features 6-degree-of-freedom (6DoF) position tracking. Underneath the Rift's fabric cover is an array of infrared micro-LEDs, which are tracked in real space by the included infrared camera. Positional tracking should always correspond 1:1 with the user's movements as long as they are inside the tracking camera's volume. Augmenting the response of position tracking to the player's movements can be discomforting.

The SDK reports a rough model of the user's head in space based on a set of points and vectors. The model is defined around an origin point, which should be centered approximately at the pivot point of the user's head and neck when they are sitting up in a comfortable position in front of the camera.

You should give users the ability to reset the head model's origin point based on where they are sitting and how their Rift is set up. Users may also shift or move during gameplay, and therefore should have the ability to reset the origin at any time. However, your content should also provide users with some means of guidance to help them best position themselves in front of the camera to allow free movement during your experience without leaving the tracking volume. Otherwise, users might unknowingly set the origin to a point on the edge of the camera's tracking range, causing them to lose position tracking when they move. This head model origin set function can take the form of a set-up or calibration utility separate from gameplay.

The head model is primarily composed of three vectors. One vector roughly maps onto the user's neck, which begins at the origin of the position tracking space and points to the "center eye," a point roughly at the user's nose bridge. Two vectors originate from the center eye, one pointing to the pupil of the left eye, the other to the right. More detailed documentation on user position data can be found in the SDK.

Room scale opens new possibilities for more comfortable, immersive experiences and gameplay elements. Players can lean in to examine a cockpit console, peer around corners with a subtle shift of the body, dodge projectiles by ducking out of their way, and much more.

Although room scale holds a great deal of potential, it also introduces new challenges. First, users can leave the viewing area of the tracking camera and lose position tracking, which can be a very jarring experience. To maintain a consistent, uninterrupted experience, the Oculus Guardian system you should provide users with warnings as they approach the edges of the camera's tracking volume before position tracking is lost. They should also receive some form of feedback that will help them better position themselves in front of the camera for tracking.

We recommend fading the scene to black before tracking is lost, which is a much less disorienting and discomforting sight than seeing the environment without position tracking while moving. The SDK defaults to using orientation tracking and the head model when position tracking is lost.

The second challenge introduced by position tracking is that users can now move the virtual camera into unusual positions that might have been previously impossible. For instance, users can move the camera to look under objects or around barriers to see parts of the environment that would be hidden from them in a conventional video game. On the one hand, this opens up new methods of interaction, like physically moving to peer around cover or examine objects in the environment. On the other hand, users may be able to uncover technical shortcuts you might have taken in designing the environment that would normally be hidden without position tracking. Take care to ensure that art and assets do not break the user's sense of immersion in the virtual environment.

A related issue is that the user can potentially use position tracking to clip through the virtual environment by leaning through a wall or object. One approach is to design your environment so that it is impossible for the user to clip through an object while still inside the camera's tracking volume. Following the recommendations above, the scene would fade to black before the user could clip through anything. Similar to preventing users from approaching objects closer than the optical comfort zone of 0.75-3.5 meters, however, this can make the viewer feel distanced from everything, as if surrounded by an invisible barrier. Experimentation and testing will be necessary to find an ideal solution that balances usability and comfort.

Although we encourage developers to explore innovative new solutions to these challenges of position tracking, we discourage any method that takes away position tracking from the user or otherwise changes its behavior while the virtual environment is in view. Seeing the virtual environment stop responding (or responding differently) to position tracking, particularly while moving in the real world, can be discomforting to the user. Any method for combating these issues should provide the user with adequate feedback for what is happening and how to resume normal interaction.

Latency

We define latency as the total time between movement of the user's head and the updated image being displayed on the screen ("motion-to-photon"), and it includes the times for sensor response, fusion, rendering, image transmission, and display response.

Minimizing latency is crucial to immersive and comfortable VR, and low latency head tracking is part of what sets the Rift apart from other technologies. The more you can minimize motion-to-photon latency in your game, the more immersive and comfortable the experience will be for the user.

One approach to combating the effects of latency is our predictive tracking technology. Although it does not actually reduce the length of the motion-to-photon pipeline, it uses information currently in the pipeline to predict where the user will be looking in the future. This compensates for the delay associated with the process of reading the sensors and then rendering to the screen by anticipating where the user will be looking at the time of rendering and drawing that part of the environment to the screen instead of where the user was looking at the time of sensor reading. We encourage developers to implement the predictive tracking code provided in the SDK. For details on how this works, see Steve LaValle's *The Latent Power of Prediction* blog post as well as the relevant SDK documentation.

At Oculus we believe the threshold for compelling VR to be at or below 20ms of latency. Above this range, users tend to feel less immersed and comfortable in the environment. When latency exceeds 60ms, the disjunction between one's head motions and the motions of the virtual world start to feel out of sync, causing discomfort and disorientation. Large latencies are believed to be one of the primary causes of simulator sickness.[1] Independent of comfort issues, latency can be disruptive to user interactions and presence. Obviously, in an ideal world, the closer we are to 0ms, the better. If latency is unavoidable, it will be more uncomfortable the more variable it is. You should therefore shoot for the lowest and least variable latency possible.

[1] Kolasinski, E.M. (1995). *Simulator sickness in virtual environments* (ARTI-TR-1027). Alexandria, VA: Army Research Institute for the Behavioral and Social Sciences. Retrieved from http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA295861

Simulator Sickness

Overview

- *"Simulator sickness" refers to symptoms of discomfort that arise from using simulated environments.*
- *Conflicts between the visual and bodily senses are the cause.*
- *Numerous factors contribute to simulator sickness, including:*
    - *Acceleration—minimize the size and frequency of accelerations*
    - *Degree of control—don't take control away from the user*
    - *Duration of simulator use—allow and encourage users to take breaks*
    - *Altitude— avoid filling the field of view with the ground*
    - *Binocular disparity—some find viewing stereoscopic images uncomfortable*
    - *Field-of-View—reducing the amount of visual field covered by the virtual environment may also reduce comfort*
    - *Latency—minimize it; lags/dropped frames are uncomfortable in VR*
    - *Distortion correction—use Oculus VR's distortion shaders*
    - *Flicker—do not display flashing images or fine repeating textures*
    - *Experience—experience with VR makes you resistant to simulator sickness (which makes developers inappropriate test subjects)*
- *Locking the background to the player's inertial reference frame has been found to be effective at reducing simulator sickness.*
- *Various methods are currently being explored for greater comfort in VR.*
- *The SSQ can be used as a means of gathering data on how comfortable your experience is.*

Description

Simulator sickness is a form of *induced motion sickness*, which differs from your everyday motion sickness. Whereas the motion sickness with which people are most familiar results from actual motion (such as the bobbing of a boat that causes seasickness), the primary feelings of discomfort associated with simulator sickness occur when visual information from a simulated environment signals self-motion in the absence of any actual movement. In either case, there are conflicts among the visual, vestibular (balance), and proprioceptive (bodily position) senses that give rise to discomfort. Furthermore, simulator sickness includes symptoms that are unique to using a virtual environment, such as eye strain/fatigue (though not necessarily for the same reason as bodily discomfort). Some users might experience some degree of simulator sickness after a short period of time in a headset, while others may never experience it.

Simulator sickness poses a comfort problem to users and developers alike. No matter how fundamentally appealing your content is or how badly a user wants to enjoy it, almost no one wants to endure the discomfort of simulator sickness. Therefore, it is extremely important to understand its causes and implement strategies to minimize its occurrence. The exact causes of simulator sickness (and in fact all forms of motion sickness) are still being researched. Simulator sickness has a complex etiology of factors that are sufficient but not necessary for inducing discomfort, and maximizing user comfort in the VR experience requires addressing them all.

Simulator sickness has a constellation of symptoms, but is primarily characterized by *disorientation* (including *ataxia,* a sense of disrupted balance), nausea (believed to stem from *vection*, the illusory perception of self-motion) and *oculomotor discomfort* (e.g., eyestrain). These are reflected in the subscales of the *simulator sickness questionnaire (SSQ)*,[1] which researchers have used to assess symptomatology in users of virtual environments.

Factors Contributing to Simulator Sickness

It can be difficult to track down a particular cause for simulator sickness. Different users will have different experiences, sensitivity to different types of stimuli can vary, and the symptoms can take a while (anywhere from minutes to hours) to manifest. As a VR designer, you will be spending long periods of time immersed in VR, and long exposure to virtual environments can train the brain to be less sensitive to their effects.[2] As such, dedicated VR developers will be less susceptible to simulator sickness than most users. Objectively predicting whether a user will experience discomfort from your content without obtaining feedback from inexperienced users can be difficult.

Motion sickness susceptibility varies in the population and correlates with the intensity of simulator sickness experiences.[3] This means users who know they tend to experience motion sickness in vehicles, rides, and other contexts should approach using VR carefully, and you should alert users to this point in your warnings and instructions. Applying the recommendations throughout this manual can help reduce the possibility that users will experience simulator sickness.

The following section lists factors that have been studied as potential contributors to simulator sickness. Some factors are less under the designer's control than others, but understanding them can help you minimize user discomfort. Also note that some of this information overlaps with other sections, but this section offers more detailed explanations of their role in simulator sickness.

Speed of Movement and Acceleration

Speed of movement is directly proportional to the speed of onset of simulator sickness, but not necessarily the subsequent intensity or rate of increase.[4] Although slower movement speeds will generally feel more comfortable, the most important issue is acceleration, which is the stimulus to which the inner ear vestibular organs respond. Acceleration (linear or angular, in any direction) conveyed visually but not to the vestibular organs constitutes a sensory conflict that can cause discomfort. An instantaneous burst of acceleration is more comfortable than an extended, gradual acceleration to the same movement velocity.

Discomfort will increase as a function of the frequency, size, and duration of acceleration. Because any period of visually-presented acceleration represents a period of conflict between the senses, it is best to avoid them as much as possible.

Note: The vestibular organs do not respond to constant velocity, so constant visual motion represents a smaller conflict for the senses.

Degree of Control

Taking control of the camera away from the user or causing it to move in ways not initiated by the user can lead to simulator sickness. Some theories suggest the ability to anticipate and control the motion experienced plays a role in staving off motion sickness,[5] and this principle appears to hold true for simulator sickness as well. Therefore, unexpected camera movement (or cessation of movement) outside the user's control can be uncomfortable. Having an avatar that foreshadows impending camera movement can help users anticipate and prepare for the visual motion, potentially improving the comfort of the experience.[6]

If you have a significant event for the user to watch (such as a cut scense or critical environmental event), avoid moving their gaze for them. Instead, encourage users to move their own gaze, for example by having non-player characters (NPCs) looking towards the scene or event, cuing them to events with sound effects, or by placing some task-relevant target (such as enemies or pick-ups) near it.

As stated previously, do not decouple the user's movements from the camera's movements in the virtual environment.

Duration

The longer you remain in a virtual environment, the more likely you are to experience simulator sickness. Users should always have the freedom to suspend their game, then return to the exact point where they left off at their leisure. Well-timed suggestions to take a break, such as at save points or breaks in the action, are also a good reminder for users who might otherwise lose track of time.

Altitude

The altitude of the user — that is, the height of the user's point of view (POV) — can be an indirect factor in simulator sickness. The lower the user's POV, the more rapidly the ground plane changes and fills the user's FOV, creating a more intense display of visual flow. This can create an uncomfortable sensation for the same reason moving up staircases, which also creates an intense visual flow across the visual field, is so discomforting.

Binocular Display

Although binocular disparity is one of the Rift's key and compelling depth cues, it is not without its costs. As described in *Binocular Vision, Stereoscopic Imaging and Depth Cues* on page 10, stereoscopic images can force the eyes to converge on one point in depth while the lens of the eye accommodates (focuses itself) to another. Although you will necessarily make use of the full range of depth in VR, it is important to place content on which you know users will be focusing for extended periods of time (such as menus or a 3rd-person avatar) in a range of 0.75 to 3.5 Unity units (meters) away.

Some people find viewing stereoscopic images uncomfortable, and research has suggested that reducing the degree of disparity between the images (i.e., reducing the inter-camera distance) to create a monoscopic[7] (i.e., zero-inter-camera distance) or microstereoscopic[8] (i.e., reduced inter-camera distance) display can make the experience more comfortable. In the Rift, it is important that any scaling of the IPD is applied to the entire head model.

As stated elsewhere, you should set the inter-camera distance in the Rift to the user's IPD from the config tool to achieve a veridical perception of depth and scale. Any scaling factors applied to eye separation (camera distance) must be also applied to the entire head model so that head movements correspond to the appropriate movements of the virtual rendering cameras.

Field of View

Field of view can refer to two kinds of field of view: the area of the visual field subtended by the display (which we call "display FOV" or dFOV in this guide), and the area of the virtual environment that the graphics engine draws to the display (which we call "camera FOV" or cFOV).

A wide dFOV is more likely to contribute to simulator sickness primarily for two reasons related to the perception of motion. First, motion perception is more sensitive in the periphery, making users particularly susceptible to effects from both optic flow and subtle flicker in peripheral regions. Second, a larger display FOV, when used in its entirety, provides the visual system with more input than a smaller display FOV. When that much visual input suggests to the user that they are moving, it represents an intense conflict with bodily (i.e., vestibular and proprioceptive) senses, leading to discomfort.

Reducing display FOV can reduce simulator sickness,[9] but also reduces the level of immersion and situational awareness with the Rift. To best accommodate more sensitive users who might prefer that compromise, you should allow for user-adjustable display FOV. Visibility of on-screen content should not be adversely affected by changing display FOV.

Having a cockpit or vehicle obscuring much of the vection-inducing motion in the periphery may also confer a similar benefit for the same reasons. Note also that the smaller the user's view of their environment, the more they will have to move their head or virtual cameras to maintain situational awareness, which can also increase discomfort.

Manipulating camera FOV can lead to unnatural movement of the virtual environment in response to head movements (for example, if a 10° rotation of the head creates a rotation of the virtual world that would normally require a 15° rotation in reality). In addition to being discomforting, this can also cause a temporary but maladaptive condition known as vestibular-ocular reflex (VOR) gain adaptation.[10] Your eyes and vestibular system normally work together to determine how much the eyes must move during a head movement in order to maintain stable fixation on an object. If the virtual environment causes this reflex to fail to maintain stable fixation, it can lead to an uncomfortable re-calibration process both inside the Rift and after terminating use.

Latency and Lag

Although developers have no control over many aspects of system latency (such as display updating rate and hardware latencies), it is important to make sure your VR experience does not lag or drop frames on a system that meets minimum technical specifications. Many games can slow down as a result of numerous or more complex elements being processed and rendered to the screen. While this is a minor annoyance in traditional video games, it can have an uncomfortable effect on users in VR.

Past research findings on the effects of latency are somewhat mixed. Many experts recommend minimizing latency to reduce simulator sickness because lag between head movements and corresponding updates on the display can lead to sensory conflicts and errors in the vestibular-ocular reflex. We therefore encourage minimizing latency as much as possible.

It is worth noting that some research with head-mounted displays suggests a fixed latency creates about the same degree of simulator sickness whether it’s as short as 48 ms or as long as 300 ms;[11] however, variable and unpredictable latencies in cockpit and driving simulators create more discomfort the longer they become on average.[12] This suggests that people can eventually get used to a consistent and predictable bit of lag, but fluctuating, unpredictable lags are increasingly discomforting the longer they become on average.

Still, adjusting to latency (and other discrepancies between the real world and VR) can be an uncomfortable process that leads to further discomfort when the user adjusts back to the real world outside of VR. The experience is similar to getting on and off a cruise ship. After a period feeling seasick from the rocking of the boat, many people become used to the regular, oscillatory motion and the seasickness subsides; however, upon returning to solid land, many of those same people will actually experience a “disembarkment sickness” as the body has to readjust once again to its new environment.[13]

The less you have to make the body adjust to entering and exiting VR, the better. Developers are urged to use the Performance HUD and Oculus Debug Tool to measure motion-to-photon latency to ensure it is as short and consistent as possible. Further documentation on its use is available in the SDK.

Distortion Correction

The lenses in the Rift distort the image shown on the display, and this is corrected by the post-processing steps given in the SDK. It is extremely important that this distortion be done correctly and according to the SDK’s guidelines and the example demos provided. Incorrect distortion can “look” fairly correct, but still feel disorienting and uncomfortable, so attention to the details is paramount. All of the distortion correction values need to match the physical device—none of them may be user-adjustable (the SDK demos allow you to play with them just to show what is happening behind the scenes).

We carefully tune our distortion settings to the optics of the Rift lenses and are continually working on ways of improving distortion tuning even further. All developers must use the official Oculus VR distortion settings to correctly display content on the Rift.

Flicker

Flicker plays a significant role in the oculomotor component of simulator sickness. It can be worsened by high luminance levels, and is perceived most strongly in the periphery of your field of view. Although flicker can become less consciously noticeable over time, it can still lead to headaches and eyestrain.

Although they provide many advantages for VR, OLED displays carry with them some degree of flicker, similar to CRT displays. Different people can have different levels of sensitivity, but the 90-hz display panels of the Rift are fast enough that the majority of users will not perceive any noticeable flicker. This is more or less out of your hands as a developer, but it is included here for completeness.

Your responsibility is to refrain from creating purposely flickering content. High-contrast, flashing (or rapidly alternating) stimuli, can trigger photosensitive seizures in some people. Related to this point, high-spatial-frequency textures (such as fine black-and-white stripes) can also trigger photosensitive seizures. The International Standards Organization has been studying photosensitive seizures and image safety and is in the process of developing a standard to reduce the risk of photosensitive seizures from image content.[14] The standard addresses potentially harmful flashes and patterns. You must ensure that your content conforms to standards and best practices on image safety.

Experience

The more experience users have had with a virtual environment, the less likely they are to experience simulator sickness.[14] Theories for this effect involve learned—sometimes unconscious—mechanisms that allow the user to better handle the novel experience of VR. For example, the brain learns to reinterpret visual anomalies that previously induced discomfort, and user movements become more stable and efficient to reduce vection. The good news is that developers should not be afraid to design intense virtual experiences for more experienced users; the bad news is that most users will need time to acclimate to the Rift and the game before they can be expected to handle those experiences.

This has a few important ramifications. First, developers who test their own games repeatedly will be much more resistant to simulator sickness than a new user, and therefore need to test the experience with a novice population with a variety of susceptibility levels to simulator sickness to assess how comfortable the experience actually is. Second, new users should not be thrown immediately into intense game experiences; you should begin them with more sedate, slower-paced interactions that ease them into the game. Even better, you should implement the recommendations in this guide for user-controlled options to adjust the intensity of the experience. Third, games that do contain intense virtual experiences should provide users with warning of the content in the game so they may approach it as they feel most comfortable.

Combating Simulator Sickness
Player-Locked Backgrounds (a.k.a. Independent Visual Backgrounds)

The simulator sickness research literature has provided at least one purely visual method of reducing simulator sickness that can be implemented in VR content. Experimenters put people in a virtual environment that either did or did not contain what they called an *independent visual background*.[15] This constituted a simple visual backdrop, such as a grid or skybox, that was visible through the simulator's primary content and matched the behavior of the stable real-world environment of the user. For example, a driving simulator might indicate movement through the environment via the ground plane, trees, and buildings passing by; however, the skybox, containing a few clouds, would remain stationary in front of the user, even when the car would turn.[16] Using a virtual environment with an independent visual background has been found to significantly reduce the experience of simulator sickness compared to a virtual environment with a typically behaving background.

This combats the sensory conflict that normally leads to discomfort by allowing the viewer's brain to form an interpretation in which the visual and vestibular senses are consistent: the user is indeed stationary with the background environment, but the foreground environment is moving around the user. Our particular implementation has used a player-locked skybox that is rendered at a distance farther away than the main environment which the player navigates. A variety of backdrops appear to be effective in our preliminary testing, ranging from realistic (a sea, horizon line, and clouded sky above) to artificial (a black, grid-lined box). As soon as the player begins any locomotion or rotation in the foreground environment with a controller or keyboard, they will notice that the distant backdrop remains stationary, locked to their real-world body's position. However, they can still look around the backdrop with head movements at any time. The overall effect is that the player feels like they are in a gigantic "room" created by the backdrop, and the main foreground environment is simply moving around them.

This method has been found to be effective in reducing simulator sickness in a variety of technologies, and the Rift is no exception. However, this method is not without its limitations. The sickness-reducing effect is contingent upon two factors: the visibility of the background, and the degree to which it is perceived as further out from the player than the foreground environment. Not all virtual environments will be outdoors or otherwise somewhere where a player-locked background will be readily visible and intuitively make sense.

These practical limitations motivated us to attempt applying our grid-lined room pattern to all virtual environments as a translucent overlay, using binocular disparity and aerial perspective (i.e., fog) as depth cues that the grid is far off in the distance. Although this generally felt effective, this can potentially reduce the user's suspension of disbelief. In addition, we found that any cues that cause the player to perceive the grid as positioned between their eyes and the foreground environment (such as making the grid opaque) abolish any benefits.

Still, employed properly, this method holds promise for allowing developers to provide a wider variety of experiences to players with less impact on comfort. Furthermore, it can also serve as a means of helping users get acclimated to the virtual environment; players might turn the locked background on when first engaging your content, then have the option to disable or attenuate the effect with time. Even the most compelling VR experience is useless if almost no one can enjoy it comfortably. Player-locked backgrounds can broaden your audience to include more sensitive users who might otherwise be unable to use your content. If an effective form of independent visual background can be implemented in your content, consider including it as a player-configurable option.

Novel Approaches

Developers have already begun exploring methods for making conventional video game experiences as comfortable in VR as they are on a computer screen. What follows are descriptions of a few of the methods we have seen to date. Although they may not be compatible or effective with your particular content, we include them for your consideration.

Because locomotion leads to vection and, in turn, discomfort, some developers have experimented with using various means of teleporting the player between different locations to move them through a space. Although this method can be effective at reducing simulator sickness, users can lose their bearings and become disoriented.[17]

Some variants attempt to reduce the amount of vection the user experiences through manipulations of the camera. An alternative take on the "teleportation" model pulls the user out of first-person view into a "god mode" view of the environment with the player's avatar inside it. The player moves the avatar to a new position, then returns to first-person view from the new perspective.

Yet another approach modifies the way users turn in the virtual environment. Rather than smoothly rotating, pressing left or right on a controller causes the camera to immediately jump by a fixed angle (e.g., 30°) in the desired direction. The idea is to minimize the amount of vection to which the user is exposed during rotation, while also generating a regular, predictable movement to prevent disorientation.

Note: All the methods described in this section have the potential of reducing discomfort at the cost of producing a veridical, "realistic" experience of the virtual environment. It is at your discretion to implement any of these methods, but keep in mind that more comfortable content will be accessible to more users and may be worth the tradeoff. A compromise between an optimally realistic and optimally comfortable experience is including these methods as user-configurable options that can be enabled or disabled. Users who experience less discomfort can opt into the more veridical experience, while sensitive users can enable methods that help them to enjoy your content.

Measurement and Testing

A wide variety of techniques have been used in the measurement and evaluation of simulator sickness. On the more technical side, indirect measurements have included galvanic skin response, electroencephalogram (EEG), electrogastrogram (EGG), and postural stability. Perhaps the most frequently used method in the research literature, however, is a simple survey: the simulator sickness questionnaire (SSQ).

Like any other questionnaire, the SSQ carries some inherent limitations surrounding the validity of people's self-reported insights into their own minds and bodies. However, the SSQ also has numerous advantages. Unlike indirect, physiological measures, the SSQ requires no special equipment or training—just a pen-and-paper and some arithmetic. Anyone can deliver the questionnaire, compute scores, and interpret those scores based on past data. For respondents, the questionnaire is short and simple, taking only a minute of time out of a playtest.

The SSQ therefore provides a lot of informational value for very little cost to the tester, and is one potential option for assessing comfort in playtesting.

[1] Kennedy, R. S., Lane, N. E., Berbaum, K. S., & Lilienthal, M. G. (1993). Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology, 3(3),* 203-220.

[2] Kennedy, R., Stanney, K., & Dunlap, W. (2000). Duration and exposure to virtual environments: Sickness curves during and across sessions. *Presence, 9(5)*, 463-472.

[3] Stanney, K. M., Hale, K. S., Nahmens, I., & Kennedy, R. S. (2003). What to expect from immersive virtual environment exposure: influences of gender, body mass index, and past experience. Human factors, 45(3), 504–20.

[4] So, R.H.Y., Lo, W.T., & Ho, A.T.K. (2001). Effects of navigation speed on motion sickness caused by an immersive virtual environment. *Human Factors, 43*(3), 452-461.

[5] Rolnick, a, & Lubow, R. E. (1991). Why is the driver rarely motion sick? The role of controllability in motion sickness. Ergonomics, 34(7), 867–79.

[6] Lin, J. J., Abi-Rached, H., & Lahav, M. (2004, April). Virtual guiding avatar: An effective procedure to reduce simulator sickness in virtual environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 719-726). ACM.

[7] Ehrlich, J.A. & Singer, M.J. (1996). Simulator sickness in stereoscopic vs. monoscopic helmet mounted displays. In: Proceedings of the Human Factors and Ergonomics Society 40th Annual Meeting.

[8] Siegel, M., & Nagata, S. (2000). Just Enough Reality: Comfortable 3-D Viewing. *IEEE Transactions on Circuits and Systems for Video Technology, 10(3)*, 387–396.

[9] Draper, M.H., Viire, E.S., Furness, T.A., & Gawron, V.J. (2001). Effects of image scale and system time delay on simulator sickness within head-coupled virtual environments. *Human Factors, 43 (1)*, 129-146.

[10] Stoffregen, T.A., Draper, M.H., Kennedy, R.S., & Compton, D. (2002). Vestibular adaptation and aftereffects. In Stanney, K.M. (ed.), *Handbook of virtual environments: Design, implementation, and applications* (pp.773-790). Mahwah, New Jersey: Lawrence Erlbaum Associates, Publishers.

[11] Draper, M.H., Viire, E.S., Furness, T.A., Gawron, V.J. (2001). Effects of image scale and system time delay on simulator sickness with head-coupled virtual environments. *Human Factors, 43(1)*, 129-146.

[12] Kolasinski, E.M. (1995). *Simulator sickness in virtual environments* (ARTI-TR-1027). Alexandria, VA: Army Research Institute for the Behavioral and Social Sciences. Retrieved from http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA295861

[13] Reason, J.T. & Brand, J.J. (1975). *Motion Sickness*. Academic Press, Inc.

[14] Welch, R.B. (2002). Adapting to virtual environments. In Stanney, K.M. (ed.). *Handbook of Virtual Environments: Design, Implementation, and Application.* Lawrence Erlbaum Associates, Publishers: Mahwah, NJ.

[15]Prothero, J.D., Draper, M.H., Furness, T.A., Parker, D.E., and Wells, M.J. (1999). The use of an independent visual background to reduce simulator side-effects. *Aviation, Space, and Environmental Medicine, 70(3)*, 135-187.

[16] Lin, J. J.-W., Abi-Rached, H., Kim, D.-H., Parker, D.E., and Furness, T.A. (2002). A “natural” independent visual background reduced simulator sickness. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *46*, 2124-2128.

[17] Bowman, D. Koller, D., & Hodges, L.F. (1997). Travel in immersive virtual environments: an evaluation of viewpoint motion control techniques,” *Proceedings of the Virtual Reality Annual International Symposium*, pp. 45-52.

User Interface

Overview

- *Heads-Up Display (HUD)*
    - *Foregoing the HUD and integrating information into the environment is ideal.*
    - *Paint reticles directly onto targets rather than a fixed depth plane.*
    - *Close-up weapons and tools can lead to eyestrain; make them a part of the avatar that drops out of view when not in use.*
- *Avatars have their pros and cons; they can ground the user in the virtual environment, but also feel unusual when discrepant from what your real world body is doing.*

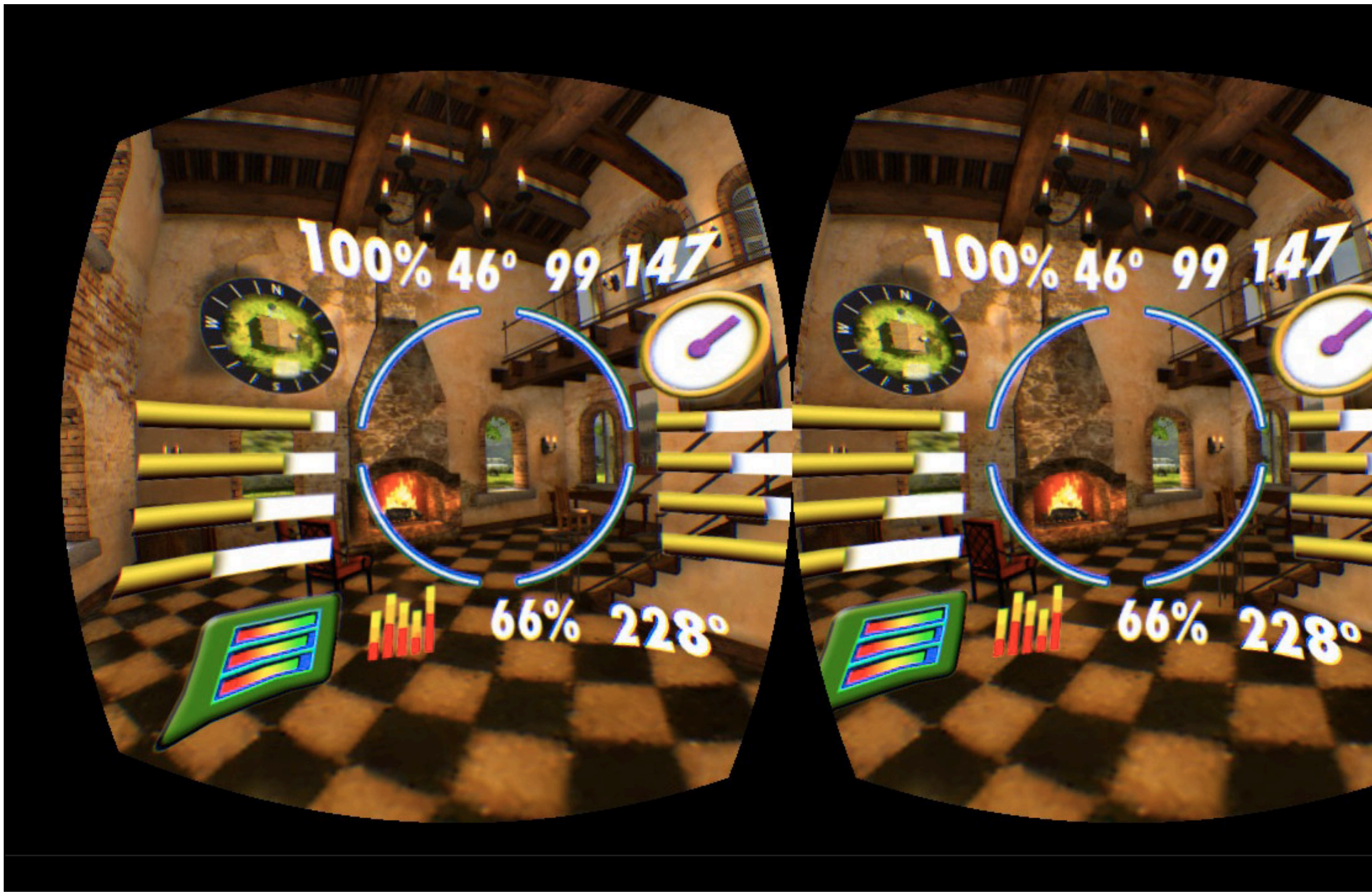
Heads-Up Display (HUD)

In general, Oculus discourages the use of traditional HUDs. Instead, we encourage developers to embed that information into the environment. Although certain old conventions can work with thoughtful re-design that is mindful of the demands of stereoscopic vision (see: reticle example below), simply porting over the HUD from a non-VR game into VR content introduces new issues that make them impractical or even discomforting.

First, HUDs *occlude* (appear in front of) everything in the 3D scene. This isn't a problem in non-stereoscopic games, because the user can easily assume that the HUD actually is in front of everything else. Unfortunately, adding binocular disparity (the slight differences between the images projected to each eye) as a depth cue can create a contradiction if a scene element comes closer to the user than the depth plane of the HUD: based on occlusion, the HUD is perceived as *closer* than the scene element because it covers everything behind it, yet binocular disparity indicates that the HUD is *farther* away than the scene element it occludes. This can lead to difficulty and/or discomfort when trying to fuse the images for either the HUD or the environment.

Although moving the HUD closer to the user might prevent visual contradictions of occlusion and disparity, the proximity necessary to prevent problems will most likely bring the interface closer than the recommended minimum comfortable distance, 75 cm. Setting the player's clipping boundary at the depth of the HUD similarly introduces issues, as users will feel artificially distanced from objects in the environment. Although they might work within particular contexts that can circumvent these issues, HUDs can quickly feel like a clunky relic in VR and generally should be deprecated in favor of more user-friendly options.

Figure 2: Busy HUD from Inside a Rift



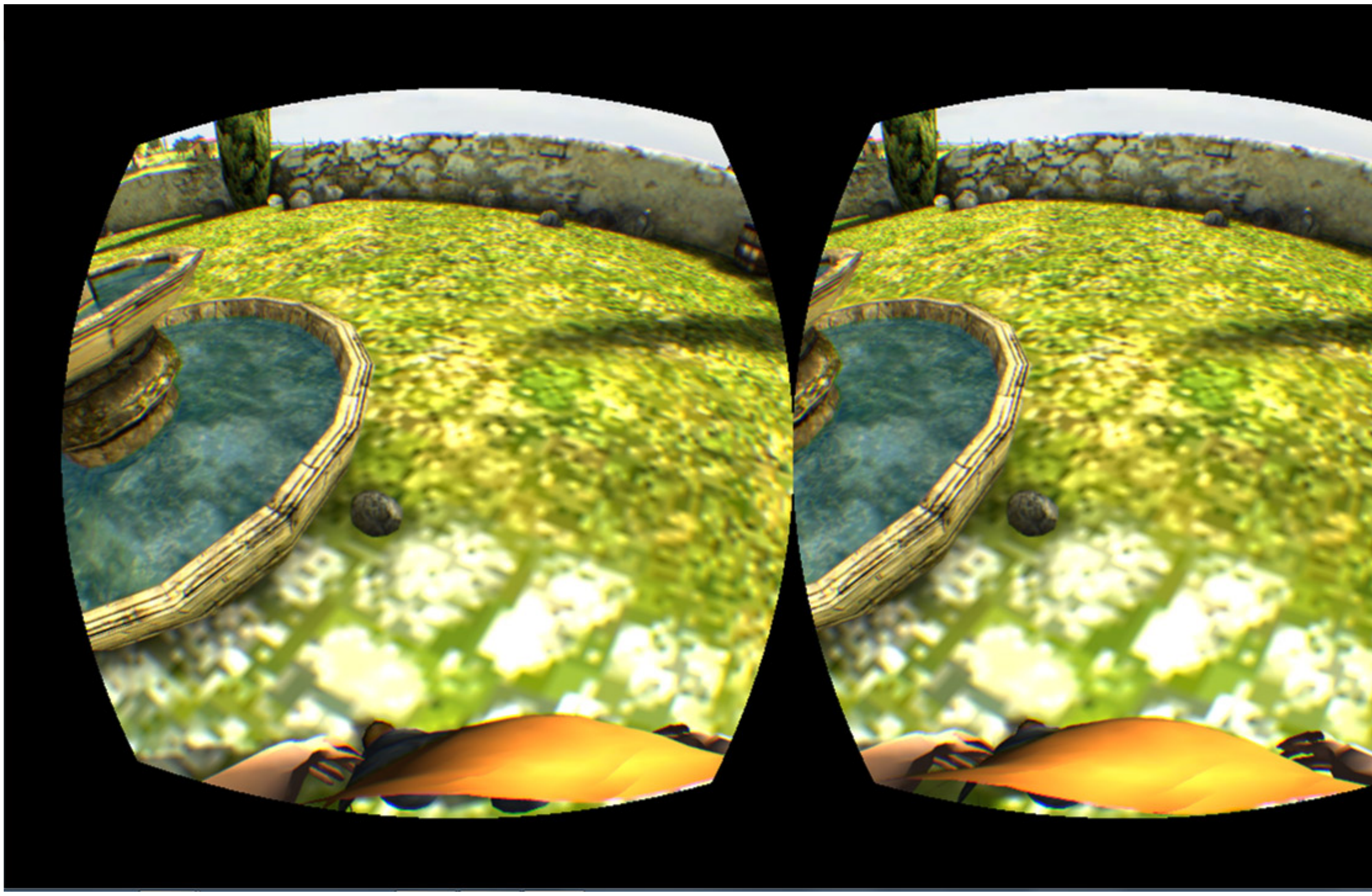
Instead, consider building informational devices into the environment itself. Remember that users can move their heads to glean information in a natural and intuitive way that might not work in traditional video games. For instance, rather than a mini map and compass in a HUD, the player might get their bearings by glancing down at an actual map and compass in their avatar's hands or cockpit. This is not to say *realism* is necessary; enemy health gauges might float magically over their heads. What's important is presenting information in a clear and comfortable way that does not interfere with the player's ability to perceive a clear, single image of the environment or the information they are trying to gather.

Targeting reticles are an excellent illustration of adapting old paradigms to VR. While a reticle is critical for accurate aiming, simply pasting it over the scene at a fixed depth plane will not yield the reticle behavior players expect in a game. If the reticle appears at a depth plane different from where the eyes are converged, it is perceived as a double image. In order for the targeting reticle to work the same way it does in traditional video games, it must be drawn directly onto the object it is targeting on screen, presumably where the user's eyes are converged when aiming. The reticle itself can be a fixed size that appears bigger or smaller with distance, or you can program it to maintain an absolute size to the user; this is largely an aesthetic decision for the designer. This simply goes to show that some old paradigms can be ported over to VR, but not without careful modification and design for the demands of the new medium.

Avatars

An *avatar* is a visible representation of a user's body in a virtual world that typically corresponds to the user's position, movement and gestures. The user can see their own virtual body and observe how other users see and interact with them. Since VR is often a first person experience, many VR applications dispense with any representation of the user whatsoever, and therefore the user is simply disembodied in virtual space.

Figure 3: User Avatar (Bottom of Screen)



Avatars have pros and cons. On the one hand, an avatar can give users a strong sense of scale and of their body's volume in the virtual world. On the other hand, presenting a realistic avatar body that contradicts the user's proprioception (e.g., a walking body while they are seated) can feel peculiar. At public demonstrations with the Rift, users generally react positively to being able to see their virtual bodies, and so avatars can serve as a means of eliciting an aesthetic response. Like anything else in this young medium, user testing and evaluation are necessary to see what works best for your experience.

Note: Since we can only bend our neck so far, the avatar's body only appears at the very edge of the image (see previous figure). Any weapons or tools should be integrated with the avatar, so the user sees the avatar actually holding them. Developers that use input devices for body tracking should track the user's hands or other body parts and update the avatar to match with as little latency as possible.

Weapons and Tools

In first person shooters, weapons typically appear towards the bottom of the screen, positioned as though the user is holding and aiming them. Spatially, this means that the weapon is much closer than anything else in the scene. In a typical non-stereoscopic game, this doesn't create any special problems, and we accept that we are seeing a big, close-up object superimposed over a scene at a normal distance.

However, when this is translated into a stereoscopic implementation, things get a little more complicated. Rendering weapons and tools so close to the camera requires the user to make large changes in eye convergence when looking between the weapon to the rest of the scene. Also, because the weapon is so close to the viewer, the left and right views can be significantly different and difficult to resolve into a single three-dimensional view.

The approach we find most comfortable is to position the cameras just above the neck of a headless, full-body avatar, as described above. Weapons and tools are rendered as part of the user avatar, which can hold them up during use, but otherwise drop them out of view.

There are some possible "cheats" to rendering weapons and tools in the player's view, and although we do not endorse them, your content might require or be suited to some variation on them. One possibility is to render weapons in 2D, behind your HUD if you have one. This takes care of some of the convergence and fusion problems at the expense of making the weapon look flat and artificial.

Another possible approach is to employ multi-rigging, so that close-up objects (e.g., cockpit, helmet, gun) are separate from the main world and independently employ a different camera separation from the environment. This method runs the risk of creating visual flaws, such as foreground objects appearing stereoscopically further away than the background behind them, and are discouraged.

Iterative experimentation and user testing might reveal an optimal solution for your content that differs from anything here, but our current recommendation is to implement weapons and tools as a component of the user's avatar.

User Input and Navigation

Overview

- *No traditional input method is ideal for VR, but gamepads are currently our best option; innovation and research are necessary and ongoing at Oculus.*
- *Users can't see their input devices while in the Rift; let them use a familiar controller that they can operate without sight.*
- *Leverage the Rift's sensors for control input (e.g., aiming with your head), but be careful of nauseating interactions between head movements and virtual motion.*
- *Locomotion can create novel problems in VR.*
- *Consider offering a "tank mode" style of movement that users can toggle. Include a means of resetting heading to the current direction of gaze.*

Mouse, Keyboard, Gamepad

It's important to realize that once users put on the Oculus Rift, they can't see their keyboard, their mouse, their gamepad, or their monitor. Once they're inside, interacting with these devices will be done by touch alone. Of course, this isn't so unusual. We're used to operating our input devices by touch, but we use sight to perform our initial orientation and corrections (such as changing hand position on a keyboard). This has important ramifications for interaction design. For instance, any use of the keyboard as a means of input is bound to be awkward, since the user will be unable to find individual keys or home position except by touch. A mouse will be a bit easier to use, as long as the user has a clear idea where the mouse is before putting on the headset.

Although still perhaps not the ultimate solution, gamepads are the most popular traditional controller at this time. The user can grip the gamepad with both hands and isn't bound to ergonomic factors of using a more complicated control device on a desktop. The more familiar the controller, the more comfortable a user will be when using it without visual reference.

We believe gamepads are preferable over keyboard and mouse input. However, we must emphasize that neither input method is ideal for VR, and research is underway at Oculus to find innovative and intuitive ways of interacting with a wide breadth of VR content.

Alternative input methods

As an alternative to aiming with a mouse or controller, some VR content lets users aim with their head. For example, the user aims a reticle or cursor that is centered in whatever direction he or she is currently facing. Internally, we currently refer to this method as "ray-casting." User testing at Oculus suggests ray-casting can be an intuitive and user-friendly interaction method, as long as the user has a clear targeting cursor (rendered at the depth of the object it is targeting) and adequate visual feedback indicating the effects of gaze direction. For example, if using this method for selecting items in a menu, elements should react to contact with the targeting reticle/cursor in a salient, visible way (e.g., animation, highlighting). Also keep in mind that targeting with head movements has limits on precision. In the case of menus, items should be large and well-spaced enough for users to accurately target them. Furthermore, users might move their heads without intending to change their target—for instance, if a tooltip appears peripherally outside a menu that is navigated by ray-casting. User testing is ultimately necessary to see if ray-casting fits your content.

The Rift sensors use information on orientation, acceleration, and position primarily to orient and control the virtual camera, but these readings can all be leveraged for unique control schemes, such as gaze- and head-/torso-controlled movement. For example, users might look in the direction they want to move, and lean forward to move in that direction. Although some content has implemented such control methods, their comfort and usability in comparison to traditional input methods are still unknown.

As a result, developers must assess any novel control scheme to ensure they do not unintentionally frustrate or discomfort novice users. For example, head tilt can seem like a reasonable control scheme in theory, but if a user is rotating in VR and tilts their head off the axis of rotation, this action creates a "pseudo coriolis effect." Researchers have found the pseudo coriolis effect to consistently induce motion sickness in test subjects,[1] and therefore should be avoided in any head-tilt-based control scheme. Similar unintended effects may exist unknowingly inside your novel input method, highlighting the need to test it with users.

Navigation

For most users, locomotion will occur through some form of input rather than actually standing up and walking around. Common approaches simply carry over methods of navigation from current gen first-person games, either with a gamepad or keyboard and mouse. Unfortunately, traditional controls—while effective for navigating a video game environment—can sometimes cause discomfort in immersive VR. For example, the simulator sickness section above described issues with strafing and backwards walking that do not affect console and PC games. We are currently researching new control schemes for navigation in VR.

Alternative control schemes have been considered for improving user comfort during locomotion. Typically, pressing "forward" in traditional control schemes leads to moving in whatever direction the camera is pointed. However, developers might also use a "tank mode" or "tank view" for navigation, where input methods control the direction of locomotion, and the user controls the camera independently with head movements. For example, a user would keep walking along the same straight path as long as they are only pressing forward, and moving their head would allow them to look around the environment without affecting heading. One might liken this to browsing an aisle in a store—your legs follow a straight path down the aisle, but your head turns side to side to look around independently of where you are walking.

This alternative control scheme has its pros and cons. Some users in the Oculus office (and presumably the developers who have implemented them in extant content) find this method of control to be more comfortable than traditional navigation models. However, this can also introduce new issues with discomfort and user experience, particularly as the direction of the user's head and the direction of locomotion can become misaligned—a user who wants to move straight forward in the direction they are looking may actually be moving at a diagonal heading just because their head and body are turned in their chair. Anyone using this method for navigation should therefore include an easy way for users to reset the heading of the "tank" to match the user's direction of gaze, such as clicking in an analog stick or pressing a button.

Further research is necessary to fully determine the comfort and effectiveness of "tank mode" under different use cases, but it represents an alternative to traditional control schemes that developers might consider as a user-selectable option.

For now, traditional input methods are a familiar and accessible option for most users, as long as developers are mindful of avoiding known issues we have described in this guide.

Some content also lends itself to alternative means of moving the player around in a virtual space. For instance, a user might progress through different levels, each of which starts in a new location. Some games fade to black to convey the player falling asleep or losing consciousness, and then have them awaken somewhere else as part of the narrative. These conventions can be carried over to VR with little issue; however, it is important to note that applying changes to the user's location in the virtual space outside their control (e.g., a jump in perspective 90° to the right, moving them to another location in the same map) can be disorienting and, depending on the accompanying visuals, potentially uncomfortable.

[1] Dichgans, J. & Brandt, T. (1973). Optokinetic motion sickness and pseudo-coriolis effects induced by moving visual stimuli. *Acta Oto-laryngologica, 76,* 339-348.

Closing Thoughts

With the Rift, you are taking unprecedented control over the user's visual reality; this presents an unprecedented challenge to developers.

The question of "What makes for effective virtual reality?" is a broad and contextual one, and we could fill tomes with its many answers. Virtual reality is still a largely uncharted medium, waiting for creative artists and developers to unlock its full potential.

As a start, VR requires new ways of thinking about space, dimension, immersion, interaction and navigation. For instance, screen-based media tends to emphasize right angles and forward motion, and the edges of the screen are always present. This leads to what cinematographers call "framing" of shots. But in VR, there is no screen, no hard physical boundaries, and there's nothing special about right angles. And there's nothing to frame, unless you use real-world elements like doorways and windows for the user to look through.

Of all forms of media, VR probably comes the closest to real world experience. Just like the physical world, it surrounds you in a completely immersive environment. You can use this to create experiences that would be impossible in any other medium. We've been sitting in front of flat screens facing forward for too long. It is more exciting and desirable than ever to leverage the space above, below, and behind the user.

Because virtual reality is a medium that attempts to replicate one's experience in the physical world, users are likely to have an expectation that they will be able to interact with that virtual world in the same ways they do outside of it. This can be a blessing and a curse: developers can use familiar real-world scenarios to guide users, but user expectations of the virtual interactions sometimes overreach the best practices for the medium. Balancing immersion, usability, and experience is just one of many challenges ahead of us in VR design.

This guide was written to provide you with the most basic foundations, critical for proper design of an engaging and comfortable VR experience. It's up to you to create the worlds and experiences that are going to make VR sing—and we can't wait for that to happen!

Be sure to visit *developer.oculus.com* for the latest information and discussions on designing VR content for the Rift.