Oculus VR Best Practices Guide

v0.008 (March 17, 2014)

For the latest version and most up-to-date information, visit
http://developer.oculusvr.com/best-practices

Authors :

Richard Yao

Tom Heath

Aaron Davies

Tom Forsyth

Nate Mitchell

Perry Hoberman

The goal of this guide is to help developers create VR that maximizes:

- ➢ ***Oculomotor Comfort - avoiding eye strain.***
- ➢ ***Bodily Comfort - preventing feelings of disorientation and nausea.***
- ➢ ***Positive User Experience - fun, immersive and engaging interactions.***

Note: As with any medium, excessive use without breaks is not recommended for you as the developer, for the end-user, or for the device.

Executive Summary of Best Practices

Rendering

- Use the Oculus VR distortion shaders. Approximating your own distortion solution, even when it "looks about right," can still be discomforting for users.
- Get the projection matrix exactly right. Any deviation from the optical flow that accompanies real world head movement creates oculomotor and bodily discomfort.
- Maintain VR immersion from start to finish – don't make a user look at static images fixed to the eyes through their Rift.
- Avoid displaying completely different content to each eye. If you have rendering effects, make sure they resolve across both eyes and do not contain random differences, as the brain will not fuse the image properly.
- Consider supersampling and anti-aliasing to remedy low apparent resolution, which is at its worst at the center of each eye.

Minimizing Latency

- Your code should run at a minimum 60fps, v-synced and unbuffered. Lag and dropped frames are discomforting in VR.
- Ideally, target 20ms or less motion-to-photon latency. Organise your code to minimize the time from sensor fusion (reading the Rift sensors) to rendering.
- Use the SDK's predictive tracking, making sure you feed in an accurate time parameter into the function call.

Optimization.

- Decrease render buffer resolution to save video memory and increase frame-rate.

Head-tracking and Viewpoint

- Avoid features 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 can be discomforting to the user.
- The display should respond to head-tracking and viewpoint changes at all times, without exception. Even when the game is paused or displaying a cutscene, users should be able to look around.
- The camera should rotate and move in a manner consistent with head movements; discrepancies are discomforting.

Accelerations

- Acceleration creates a mismatch between your visual and vestibular 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*. 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.

Speed

- Users are most comfortable moving through virtual environments at real-world speeds; a walking rate of 1.4 m/s is most comfortable.
- 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.

Cameras

- Subjecting the user to any change in perspective can induce uncomfortable feelings of movement and acceleration. Even a seemingly trivial movement—such as shifting the camera slightly to center it over a gun's sight when entering an aiming mode—can be disorienting and uncomfortable.
- Zooming in or out with the camera can induce or exacerbate simulator sickness, particularly if it causes camera movement rates to differ from head movements.
- For third-person content, be aware that camera accelerations and movements can induce nausea regardless of what your avatar is doing. Furthermore, users must always have the freedom to look around the environment, which can add new requirements to the design of your content.
- Apply the Oculus head model to create an accurate and comfortable visual frame of reference.
- 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"; it creates a series of small but uncomfortable vertical 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 much 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.
- 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.

In-game Impacts and Collisions

- Do not move the camera without the user's control (even brief shaking or jerking) during impacts and collisions. It represents unexpected, uncontrolled changes to acceleration, orientation or rotation, which creates discomfort.
- Consider settings for user-adjustable camera behavior; lower settings would not allow impacts and collisions to affect the camera, whereas higher settings would.

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

- For individualized realism and a correctly scaled world, set left and right eye camera separation to the IPD from the user's profile. Note that realism has its downside: Beyond a fairly close range, you will perceive little stereoscopic 3D. Resist the temptation to increase the inter-camera distance to enhance the stereoscopic depth effect.
- Avoid placing static visual elements that persist in the user's view (such as a HUD) closer than 50 cm from the user in the virtual world. Converging the eyes on such close objects can cause eye strain and even make clearly rendered objects appear blurry. Users might still choose to position themselves closer to environmental objects, but you should avoid forcing them into such situations when possible.

User Interface

- UIs should be a 3D part of the virtual world and ideally sit at least 50 cm 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. Your UI should fit entirely inside the middle 1/3rd of the screen; otherwise, they should be able to examine it 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.
- Consider having 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 blurry and/or doubled image when it is not at the plane of depth on which the eyes are focused.
- In general, avoid requiring the user's eyes to make rapid and frequent adjustments in distance, such as switching focus between a distant object and nearby HUD element.

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 orientation 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.

Content

- Give the user a body. Looking down and having no body is disconcerting, whereas anything from a vague, ghostlike presence to a full character avatar can do a lot to ground the user in the virtual environment.
- Consider the size and texture of your artwork as you would with any system where visual resolution is an issue (e.g. avoid very thin objects).
- Unexpected vertical accelerations outside the user's real world movements, like walking over uneven or undulating terrain, can create discomfort. Consider flattening your walking 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 so they can decide whether or not they will be able to handle it.
- In VR, simply looking at interesting shapes and textures can be a fascinating experience. Content that is simply window-dressing when played on a monitor can itself be a major focus of VR.
- 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.
- Design environments and interactions to minimize the need for strafing, back-stepping, or spinning.
- Steady, forward movement is the most comfortable for users in a virtual environment.
- People will typically move their heads/bodies if they have to shift their gaze to 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 strain.
- 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).

Health and Safety

- Carefully read and implement the warnings that accompany the Rift (Appendix L) to ensure the health and safety of both you, the developer, and your users.

Appendices for further reading and detail

1. Appendix A - Introduction to Best Practices
2. Appendix B - Field of View and Scale - 0.2 SDK version
3. Appendix C - Binocular Vision, Stereoscopic Imaging and Depth Cues
    a. Basics
    b. Monocular depth cues
    c. Ergonomics and potential issues to avoid
4. Appendix D - Rendering Techniques
    a. Display resolution
    b. Rendering resolution
    c. Dynamically-rendered impostors
5. Appendix E - Motion
    a. Speed of Movement and Acceleration
    b. Degree of Control
    c. Head Bobbing
    d. Forward and lateral movement
6. Appendix F - Tracking
    a. Orientation Tracking
    b. Position Tracking
    c. Latency
7. Appendix G - Simulator Sickness
    a. Description
    b. Theories of Simulator Sickness
    c. Factors Contributing to Simulator Sickness
    d. Speed of Movement and Acceleration
    e. Degree of Control
    f. Duration
    g. Altitude
    h. Binocular Display
    i. Field of View
    j. Latency and Lag
    k. Distortion Correction
    l. Flicker
    m. Age & Motion Sickness Susceptibility
    n. Experience
8. Appendix H - User Interface
    a. Porting 2D Elements
    b. Heads-Up Display (HUD)
    c. Avatars
    d. Weapons and Tools
9. Appendix I - User Input and Navigation
    a. Mouse, Keyboard, Gamepad
    b. Alternative input methods
    c. Navigation

10. Appendix J - Content Creation
    a. Novel Demands
    b. Art Assets
    c. Audio Design
    d. User and Environment Scale
11. Appendix K - Closing thoughts on effective VR (for now)
12. Appendix L - Health and Safety Warnings

Appendix A - Introduction to Best Practices

- *This guide will help you make comfortable, usable VR content. Visit* http://developer.oculusvr.com/best-practices *for the most up-to-date information.*

These appendices serve to elaborate on the best practices summarized above for producing Virtual Reality (VR) experiences for the **Oculus Rift**. *Best practices* are methods that help provide high quality results, and are especially important when working with an emerging medium like VR. Overviews and documentation for the Oculus SDK and integrated game engine libraries (such as **Unity, Unreal Engine, and UDK**) can be found at http://developer.oculusvr.com.

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. Enabling the mind's continual suspension of disbelief requires particular attention to detail. It can be compared to the difference between looking through a framed window into a room, versus walking through the door into the room and freely moving around.

The Oculus Rift is the first VR system of its kind: an affordable, high-quality device with a wide field of view and minimal lag. Until now, access to VR has been limited primarily to research labs, governments, and corporations with deep pockets. With the Oculus Rift, developers, designers, and artists are now leading the way toward delivering imaginative realms to a global audience.

If VR experiences ignore fundamental best practices, they can create simulator sickness—a combination of 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, poorly 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 us to make authoritative statements. In these cases, we put forward informed theories and observations and indicate them as such. User testing is absolutely crucial for designing engaging, comfortable experiences; VR as a popular medium is still too young to have established conventions on which we can rely. We count on you, the community of Oculus Rift developers, to provide feedback and help us mature these evolving VR best practices and principles.

Appendix B - Field of View and Scale - 0.2 SDK version

- *Real-world and virtual FOV (abbreviated cFOV and dFOV here) need to match. In general, don't mess with the default FOV.*

For the sake of precision, we must first disambiguate different uses of the term "field of view." We will use the term *display field of view* (dFOV) to refer to the part of the user's visual field occupied by VR content. It is a physical characteristic of the hardware and optics. The other type of type of FOV is *camera field of view* (cFOV), which refers to the range of the virtual world that is seen by the rendering camera 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). This works because the image on the monitor is simply a two-dimensional object that is seen within the observer's total view of the environment. While the image may be compelling, it does not occupy the user's entire visual world, and differences between cFOV and dFOV do not cause problems for most people.

In virtual reality, there is no view of the external room; the virtual world fills a large section of your field of view and provides the only visual feedback to your head movements. 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.

Deviations between dFOV and cFOV can be extremely discomforting.[1] The scale of the scene will no longer appear to correspond to real-world dimensions, the motion of the head and the motion of the eyes will no longer match, and the default distortion correction values will cause the rendered scene to warp. Manipulating the camera FOV can induce simulator sickness and can lead to a maladaptation in the vestibulo-ocular reflex, which allows the eyes to maintain stable fixation on an object during head movements.

You should not change any of the default settings related to FOV or scale under any circumstances, and care must be taken to get the view scaling exactly the same way the SDK specifies. As noted elsewhere, it is quite easy to get a view that looks almost—but not exactly—right, causing disorientation and discomfort in users.

[1] 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.

Appendix C - Binocular Vision, Stereoscopic Imaging and Depth Cues

- *The brain uses differences between your eyes' viewpoints to perceive depth.*
- *Set the distance between the virtual cameras to the distance between the user's pupils from the OVR config tool.*
- *Don't neglect monocular depth cues, such as texture and lighting.*
- *Objects are most comfortable to look at in the range of 0.5 to 2 m from the viewer in real life; apply the same rule of thumb to VR.*
- *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 bad.*

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 our eye's different viewpoints of the physical world, 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 *inter-pupillary distance* (IPD), and we refer to the distance between the two virtual cameras 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 4000 U.S. Army soldiers) is about 63.5 mm—the same as the Rift's *interaxial distance (IAD)*, the distance between the centers of the Rift's lenses (as of this revision of this guide). In general, the ICD should match the user's IPD (and NOT the interaxial distance) so that the virtual world is scaled correctly to the user (see figure 1).

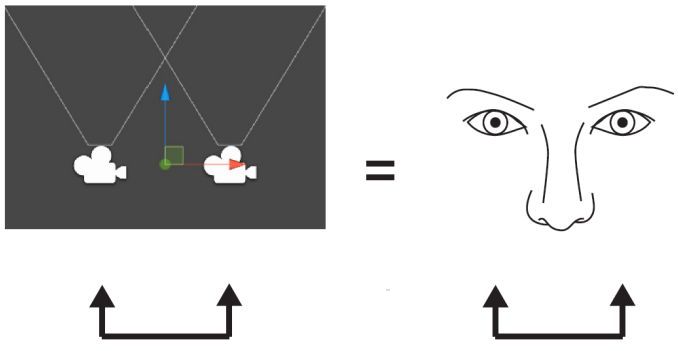

Figure 1: The inter-camera distance (ICD) between the left and right scene cameras (left) should be equal to the user's inter-pupillary distance (IPD; right).

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 is extremely important. 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 2D 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.

Ergonomics and potential issues to avoid

Humans are most comfortable focusing on objects that are approximately 0.5 to 2 meters away from them (though the ideal distance varies from person to person). The closer objects are to the viewer, the more different the two views become, and the more the eyes have to *converge* (i.e., point inward) to look at those objects. Eventually, close objects become difficult or impossible to fuse into a single image, and attempts to do so can quickly become uncomfortable.

Another reason to avoid making users focus on such close objects is the *accommodation-vergence reflex,* which causes the lens of your eye to *accommodate* (i.e., change its focus) to the point where the two eyes converge. However, because the Rift's optics simulate an image displayed at a fixed distance that does not necessarily match the point of convergence, the eyes can have difficulty accommodating to focus properly on the object, leading to eyestrain and discomfort.

Obviously, we have no control over a user who chooses 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. However, for objects that developers know will persist in the user's field of view (such as HUD elements or parts of the avatar, like a weapon or hands), these objects should be rendered at least 50 cm from the camera to maximize comfort. Also consider implementing user-adjustable settings when appropriate, such as adjusting the depth plane of a HUD to the user's preferred distance.

Changing inter-camera distance can also 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 condition 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. Therefore, we advise against artificially increasing inter-camera distance past the user's actual IPD.

The second effect of changing the inter-camera distance is that it can affect the user's sense of scale. For example, increasing the inter-camera distance to large values outside human proportions (e.g., 1 meter) can make the virtual world appear "miniaturized," as it creates a degree of binocular disparity that humans would only encounter when looking at small objects up close. We are currently working on implementations of this effect for future versions of the SDK, but until we better understand how to achieve this effect without compromising user comfort, we advise against attempting it. Similarly, we advise against experimenting with hypostereo (i.e., setting the ICD to a number below the user's physical IPD), as it introduces the same types of issues.

It is also important to ensure that the images between the two eyes do not differ aside from the slightly different viewing angles that define binocular disparity. Large differences will be obvious and make you feel like you're looking at the image cross-eyed; however, smaller differences can be problematic as well. For example, you must make sure any rendering effects (such as light distortion or particle effects) appear in both eyes and differ only in viewing angle. Failure to do so will give the final image an uncomfortable flickering/shimmering appearance to the user.

Appendix D - Rendering Techniques

Display resolution

- *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.*

The display panel used in the first Oculus Rift Development Kit (DK1) has a resolution of 1280 x 800, with an approximate FOV of 100° (exact numbers depend on the user's eye-relief and IPD). This results in approximately 6 pixels per degree of visual angle. As display technologies improve, so will the Oculus Rift, and future models will deliver a higher resolution experience.

Because the individual pixels can be seen, it creates an artifact known as the *screen door effect* — the image appears to be overlaid with a black grid pattern (see Figure 2). This pattern is in fact the space between the individual pixels. This results in aliasing artifacts, and a loss of fine detail in distant objects. Although higher resolutions will alleviate this problem somewhat, the pixel density of current display technologies cannot yet eliminate it completely.

Figure 2: The "screen door" effect.

Rendering resolution

The Rift has a display resolution of 1280x800, but before a standard projectively-rendered scene can be placed on the display, it must go through a distortion process to counteract the effects of the lenses. This process is described in detail in the SDK. It is important to realize that the process of projection and distortion changes the pixel density at different places on the image - in particular, it squashes the edges of the image (and the lens then expands the edges before presenting the image to the eye). This concentrates the limited resolution of the Rift in the center where visual detail is most needed. This means that to render a 3D scene using standard linear-projection

that matches the pixel-density of the Rift's display at the center, the standard scene usually needs to be quite a lot larger than the physical display of the Rift.

The SDK handles most of the details, but it is important to understand that the physical resolution of the Rift hardware is not directly related to the size of the target that the scene is rendered to. As a concrete example, with an eye relief of 10mm and a horizontal FOV of 100°, to match the density of the Rift's display density at the center needs an image 910 pixels wide for each eye—a total of 1820 pixels wide. This is significantly wider than the physical size of the display, which is 1280 pixels— this is a direct result of the way the lenses distort the image.

Such large render targets can be a performance problem for some graphics cards, and dropping framerate produces a poor VR experience, so an application may choose to drop resolution to improve performance. Rather than drop the resolution of the signal sent to the Rift, it is better to drop the resolution of the texture used to render the scene, and then let the distortion processing scale that up or down as necessary as part of the distortion process. This preserves as much visual fidelity as possible while increasing performance. This process is covered in more detail in the SDK.

Dynamically-rendered impostors

This method can reduce the computational burden of rendering the whole scene twice (once for each eye). By rendering sections of the scene onto textures, you can incur the majority of the render cost just once, and then subsequently render a polygon with that texture in each of the left and right eye images. As a whole, the "impostor" is placed stereoscopically at the correct distance from the eyes, but without variation within the impostor itself. This technique is therefore best applied at a suitable distance from the camera (at least 6 meters), so that the lack of depth variation within the impostor does not become noticeable.

This method can work on large swathes of general scenery as well as discrete objects, as long as sufficient care is taken at the interfaces between impostor sections.

Appendix E - Motion

- *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

Speed of movement is 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), at least as a user-configurable option.

For VR content, acceleration is a larger problem. This is because the main unit of analysis for the human vestibular system is acceleration, and perceiving acceleration without actually applying acceleration to your head or body creates discomfort. (See our section on simulator sickness for a more detailed discussion of why.)

Keep in mind that "acceleration," from a physics standpoint, 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.

Informal testing in the Oculus VR offices suggests that 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. Some exceptions may exist, but 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 the user 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 is sickening in VR. Similarly, do not freeze the display so that

[2] 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

it does not respond to the user's head movements. In general, avoid decoupling the user's and camera's movements for any reason.

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 it 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 cause discomfort. We advise against using 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, forward user movement is most comfortable, and provides depth cues that work together with the stereoscopic depth in the scene. The viewer tends to focus on the center of the scene, which changes evenly and gradually, while the periphery moves more rapidly.

Left or right lateral movement is more problematic because we don't normally walk sideways and because the viewer's focus has to hop from object to object as the scene pans. 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 very discomforting for many people. This sensitivity seems to be related to *vection* (the visually-induced perception of self-motion without actually moving), a major underlying cause of simulator sickness.[3] 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. Users do not typically see imagery like this except for rare situations like looking directly at a textured wall or floor while walking alongside. We recommend that developers use slopes and stairs sparingly. This warning 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 strongly 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 doing so forces users to engage in relatively more motions (i.e., turning, moving forward, and turning again) to accomplish the same changes in position. Besides affecting user experience, extraneous movements can lead to more simulator sickness more than an

[3] 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.

efficient alternative that might superficially seem to take control away from the user[4] (such as a complex maneuver for navigating obstacles). 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.

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

Appendix F - Tracking

- *The Rift sensors collect information about user yaw, pitch, and roll. Position tracking will be available in the future.*
- *Implement the "head model" code available in our SDK demos for camera movement.*
- *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, a consistent, fixed latency is better than a variable one*

Orientation Tracking

The Oculus Rift hardware includes 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 DK1 Rift currently tracks the user's *orientation* (rotation of the head) but not their *position*. An object in three-dimensional space has *six degrees of freedom* (x, y and z position as well as x y and z rotation), and to fully describe its *pose* we need to know both its orientation *and* its position.[5] Without additional hardware for the Rift to support positional tracking, a user is limited to adjusting positional data through a mouse or gamepad. This makes it difficult to naturally represent user actions like looking around a corner in VR. Future versions of the Rift will track the user's position out of the box.

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

---

[5] Fisher, S., McGreevy, M., Humphries, J., & Robinett, W. (1986). Virtual environment display system. *Proceedings 1986 ACM Workshop on Interactive 3D Graphics,* 77–87.

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 serve to 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 blog post at http://www.oculusvr.com/blog/the-latent-power-of-prediction/ 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.[6] In any case, latency can be disruptive to user interactions and presence. Obviously, in an ideal world, the closer we are to 0ms, the better.

Because the world is not ideal, however, we should also note that some research has suggested that the research literature on latency is mixed. Studies that report greater discomfort as a result of longer latencies tend to employ latencies that can vary within some margin of error[7]; on the other hand, studies where latency is consistent and fixed show no increase in discomfort with increases in latency.[8] This suggests that people can eventually get used to a consistent and predictable bit of lag, but fluctuating, unpredictable lags are always discomforting.

---

[6] 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

[7] E.g., Uliano, K.C., Lambert, B.Y., Kennedy, R.S., & Sheppard, D.J. (1986). *The effects of asynchronous visual delays on simulator flight performance and the development of simulator sickness symptomatology* (AD-A180 196). Alexandria, VA: Army Research Institute for the Behavioral and Social Sciences. Retrieved from http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA180196

[8] 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.

Appendix G - Simulator Sickness

- *"Simulator sickness" refers to symptoms of discomfort that arise from using simulated environments.*
- *Conflicts between the visual and bodily senses are to blame.*
- *Numerous factors contribute to simulator sickness, including but not limited to*
    - *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: a narrower FOV may help prevent discomfort*
    - *Latency: minimize it—lags/dropped frames are uncomfortable in VR*
    - *Distortion correction: use Oculus VR's distortion shaders*
    - *Flicker: do not display flashing images*
    - *Age: It is not yet clear how much age predicts simulator sickness*
    - *Experience: experience with VR makes you resistant to simulator sickness (which makes developers the worst test subjects)*

Description

Simulator sickness is a form of *visually induced motion sickness*, which differs crucially 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 between the visual and vestibular senses. Furthermore, simulator sickness includes symptoms that are unique to using a virtual environment, such as eye strain/fatigue. Some users will 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 serious problem to users and developers alike; no matter how fundamentally appealing your content is or how badly a user wants to enjoy it, 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. Unfortunately, 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 truly "curing" it requires addressing them all.

Adding to the complexity is the novelty of the Rift. While we do have some understanding of the fundamental contributors to simulator sickness, the Rift's technology and use case have no direct analog in the extant research literature.

Simulator sickness has a number 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 as the result of the oculomotor system working overtime to correct errors). These are reflected in the subscales of the *simulator sickness questionnaire (SSQ)*,[9] which researchers have used to assess symptomatology in users of virtual environments.

Theories of Simulator Sickness

The most widely accepted theory of simulator sickness is *sensory conflict theory*, which holds that a mismatch between visual and body-based (vestibular and proprioceptive) motion cues causes discomfort.[10] In other words, vision is telling users they are moving, while the body is saying they are sitting still. This is, of course, a fundamental quality of current VR; for instance, if we want to represent the experience of walking, we are forced to do it visually; the user won't actually be walking. There are active attempts in the industry to provide the appropriate body cues through motion platforms and treadmills, though results are mixed and in some cases introduce new issues.

There are two competing theories that give alternative explanations of simulator sickness. First, the *poison theory* posits that ingesting poison causes conflicts in the vestibular and visual systems. The theory argues that nausea and vomiting as a result of sensory conflict is evolutionarily adaptive because it would rid the stomach of the toxin and condition the animal to avoid whatever item presumably contained the poison it ingested. Forms of motion sickness are theorized to be the result of the user's body believing they've been poisoned due to the conflicting information they experience, and reacts accordingly.[11]

Second, the *postural instability theory* posits that the prolonged loss of postural control leads to simulator (and motion) sickness, since the subject is receiving conflicting information that causes usual postural control strategies to fail.[12] Although the theory is agnostic to *why* instability would lead to sickness, it is useful for predicting when motion and simulator sickness might occur and to what severity.

Note that regardless of the underlying theory, increasing the correspondence among the user's visual, proprioceptive, vestibular, and motor systems should reduce or eliminate simulator sickness.

---

[9] 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.

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

[11] Treisman, M. (1977). Motion sickness: An evolutionary hypothesis. *Science, 197*, 493-495.

[12] Riccio, G.E. & Stoffregen, T.A. (1991). An ecological theory of motion sickness and postural instability. *Ecological Psychology, 3(3)*, 195-240.

Factors Contributing to Simulator Sickness

It can be difficult to track down a particular cause for simulator sickness; different users will have different experiences, 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. As such, we expect dedicated VR developers and users to be less susceptible than most users. The down-side is that objectively predicting whether a user will experience discomfort from your content without obtaining feedback from inexperienced users can be problematic.

Motion sickness susceptibility is widely variable in the population and correlates with the intensity of subsequent simulator sickness experiences.[13] This means users who know they tend to experience motion sickness in vehicles, rides, and other contexts should approach using the Rift carefully. Applying the recommendations throughout this manual can help.

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 elaborated explanations for their role in simulator sickness.

Speed of Movement and Acceleration

Speed of movement is directly proportional to the speed of *onset* for simulator sickness, but not necessarily the subsequent intensity or rate of increase.[14] Although slower movement speeds will generally feel more comfortable, the real enemy to beware is acceleration, which is the unit of analysis of the vestibular organs inside the inner ear. Acceleration (linear or angular, in any direction) conveyed visually but not to the vestibular organs constitutes a sensory conflict that can cause discomfort. Informal testing suggests instantaneous 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 acceleration represents a period of conflict between the senses, it is best to avoid them as much as possible.

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 cause simulator sickness. Some theories suggest the ability to

---

[13] 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.

[14] 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.

anticipate and control the motion experienced play a role in staving off motion sickness,[15] 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.

If you have a significant event for the user to watch (such as a cutscene or critical environmental event), avoid moving their gaze for them; instead, try to provide suggestions that urge them to move gaze themselves, for example by having non-user characters (NPCs) looking towards it, 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.

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 themselves in your game.

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. Under certain conditions, stereoscopic images can force the eyes to converge on one point in depth while the lens of the eye accommodates (focuses itself) to another. This is is why we discourage forcing the user to look at close objects or visual elements for a prolonged period of time (such as reading a HUD).

Some people find viewing stereoscopic images uncomfortable, and some research has suggested monoscopic displays can even be more comfortable than stereoscopic ones.[16] Reports from some early Rift users, however, say the opposite. Further research must be conducted before we can make any concrete recommendations for or against any particular monoscopic implementation. The issue is further complicated by the dynamics of how the displayed image should react to head movements, as they will differ for the left and right eyes. To be safe, developers should avoid subjecting users to any

[15] 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.

[16] 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.

experiments in monoscopic presentation for now.

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), and the area of the virtual environment that the graphics engine draws to the display (which we call "camera FOV" or cFOV).

A wide display FOV is more likely to cause 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 starts conflicting with bodily (i.e., vestibular and proprioceptive) senses, leading to discomfort.

Reducing display FOV can reduce the experience of simulator sickness,[17] 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 HUDs and other on-screen information should not be adversely affected by these apertures, and could even be displayed outside the display FOV containing VR content. Having a cockpit or vehicle obscuring much of the vection-inducing motion in the periphery may also confer a similar benefit, but more extensive testing is necessary to determine the ideal characteristics of cockpits and the degree to which they actually help.

Unnatural movement of the virtual environment in response to head movements (for example, if a 5° rotation of the head creates a rotation of the virtual world that would normally require a 10° rotation in reality) can also cause a temporary but maladaptive condition known as vestibulo-ocular reflex (VOR) gain adaptation.[18] 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

Past research findings on the effects of latency are 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 vestibulo-ocular reflex. 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; however, variable and unpredictable

---

[17] 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.

[18] 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.

latencies in cockpit and driving simulators create more discomfort the longer they become on average.[19]

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 specification requirements. 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 intensely uncomfortable effect on users in VR.

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 is done correctly and according to the SDK's guidelines and the example demos provided. Our eyes are extremely sensitive to variations here, and 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, not because this is a particularly sensible thing to do).

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 are expected to use the official Oculus VR distortion settings to correctly display content on the Rift.

Flicker

Flicker can generally be noticed at refresh rates lower than 60 hz on CRT and OLED displays; however, the Rift DK1 display is a 60hz LCD, and should not contain perceivable flicker. But, be aware that flicker plays a significant role in simulator sickness, and it can be unpleasant, tiring and distracting. 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. 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, particularly in the 1-20 Hz range, can trigger seizures in people with photosensitive epilepsy.

Age & Motion Sickness Susceptibility

It is not yet clear how strongly age-related susceptibility to motion sickness predicts susceptibility to simulator sickness. Experience with the real world analog of a virtual

---

[19] 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

task has been positively correlated with simulator sickness,[20] suggesting experienced adults would be more susceptible than inexperienced children in a realistic simulation. Anecdotal reports from DK1 Rift developers have remarked on an apparent resilience among children and adolescents compared to adults. On the one hand, this might suggest age-related susceptibility to motion sickness does not translate to simulator sickness. On the other hand, these stories do not account for alternative explanations, such as a lower threshold for adults in deciding to abandon the Rift or a lower awareness in children of bodily discomfort.

In one study of over a thousand men and women age 15-53, an individual's personal history of motion sickness susceptibility was correlated with the intensity of simulator sickness experienced from a virtual environment, though no age effects were reported.[21] This suggests a particular individual's history of motion sickness, perhaps regardless of age, may help predict how well they will handle VR.

Experience

The more experience a user has had with a virtual environment, the less likely they are to experience simulator sickness.[22] 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.

---

[20] Ibid.

[21] 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), 504-520.

[22] 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.
Prothero, J.D., Draper, M.H., Furness, T.A., Parker, D.E., & Wells, M.J. (1999). The use of an independent visual background to reduce simulator side-effects. *Aviation, Space, and Environmental Medicine, 70(3.1)*, 277-283.

Appendix H - User Interface

- *The StereoConfig helper class can help port 2D content to VR*
- *Heads-Up Display (HUD)*
    - *Make HUDs a 3D part of the environment*
    - *Paint reticles directly onto targets rather than a fixed depth plane*
    - *Foregoing the HUD and integrating information into the environment would be ideal*
- *Close-up weapons and tools can lead to eye strain; make them a part of the avatar that drops out of view when not in use.*
- *Give users an avatar to ground them in the virtual environment.*

Porting 2D Elements

For quick ports of 2D data (main menus, UI elements, etc.), the StereoConfig helper class creates a projection matrix that puts a virtual 2D layer about 0.8m in front of the user. This is primarily intended for getting UIs implemented quickly, not for final polished interfaces.

Heads-Up Display (HUD)

A problem comes up when integrating HUD elements with stereoscopic imagery, because HUDs *occlude* (appear in front of) everything in the 3D scene. This isn't a problem in non-stereoscopic games, because the user simply assumes 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.

A common first attempt at a solution is to create the HUD very close to the user, perhaps 20cm away (contradictory to our recommended, comfortable range of 50 to 200 cm). This can reduce the frequency of cue conflicts, but now the scene itself is far behind the HUD in stereoscopic space. This requires the user to shift their focus between the close-up HUD and the much more distant scene whenever they check the HUD. These kinds of shifts in eye convergence and accommodation (eye lens focus) can quickly lead to fatigue and eyestrain.

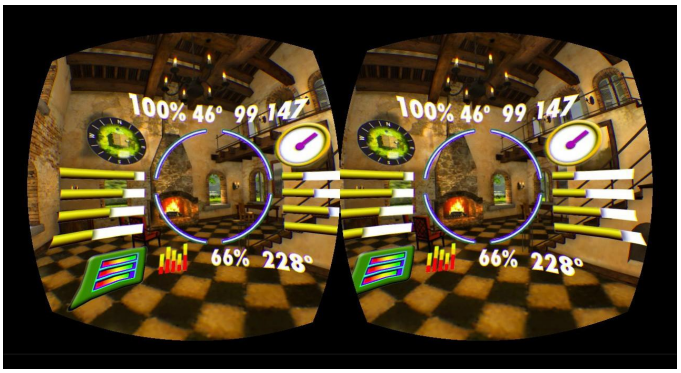

Figure 3: Example of a very busy HUD rendered as though it appears on the inside of a helmet visor.

There are a number of possibilities to properly integrate a HUD into a stereoscopic scene. For example, collision detection might prevent users from getting closer to any object than the distance between them and the HUD, preventing any clipping. However, that can also make interaction with the environment feel unnatural when users cannot approach objects as closely as expected.

Another possibility would be to draw the HUD elements not as an overlay, but as 2D surfaces in the 3D scene. Then, when objects in the 3D scene come closer than the HUD, they will occlude it. This looks natural, but you run the risk of temporarily obstructing the HUD from the user's view.

Although the elements in a HUD are 2D, the HUD itself should be a 3D object in your scene. We've found that one of the most effective strategies is to render the HUD onto a transparent hemisphere, as though it's appearing on the inside of a helmet visor (see figure 3). This can integrate the HUD into the scene itself in a believable and realistic fashion, though you will have to assess the applicability of this approach against the visual and thematic aesthetic of your VR content.

Targeting reticles likewise present new challenges in stereoscopic VR. The reticle itself should be drawn directly onto objects in the scene so that it appears where users' eyes are going to be focused when aiming. If the reticle appears at a depth plane different from where the eyes are accommodated and converged, it is perceived as blurry and doubled. 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 (though its changing size can serve as a depth cue to users).

Painting the aiming cursor directly onto objects has inspired some developers to also place the HUD at the same distance plane as the reticle. Unfortunately, the larger visual size of the HUD means the rapid changes in apparent depth are very disorienting.

Generally, the closest objects in a scene are in the lower part of the screen (objects, furniture, people) while the upper part of the screen contains relatively distant objects like ceilings and skies. You might consider locating most of your HUD elements in the upper part of the screen.

We recommend limiting your use of HUDs. Instead, consider building informational devices into the scene itself. When HUDs must be used, you must consider issues such as depth cue conflicts, clipping, and collision within your VR content.

If you do use a HUD, include options for user adjustments. Some users exhibit strong preferences regarding the distance of the HUD. Some find it difficult to switch between the world and the HUD, and so want it several meters away from them, even if this means it seems to be "behind" many elements of the game world. Other users dislike the HUD getting in the way of the environment and prefer it close to them at a distance that causes some people eyestrain. There appears to be no universally ideal distance, so we strongly advise having an easy way to quickly adjust the HUD distance between 20cm and 20m in logarithmic steps.

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.

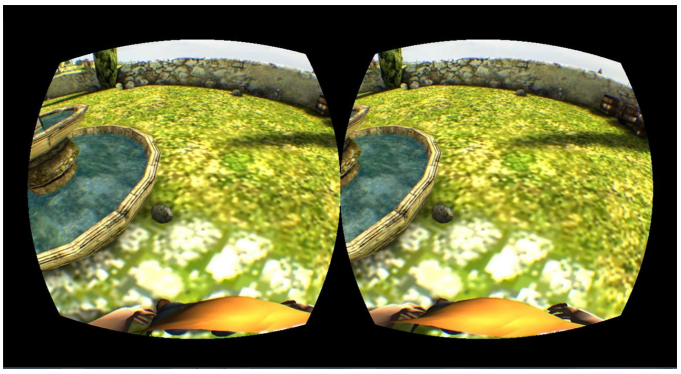

Figure 4: A user avatar, seen at the bottom of the screen.

Unless you're making a game where the user is invisible, you should give the user a visible avatar. It can be anything from a vague, ghost-like representation of where their body is to a fully fleshed-out, realistic body. This can help ground the user in the virtual space. Because we can't see our own heads, the avatar should be headless (except when seen in reflections, of course), with the bottom of the virtual neck positioned just below the camera.

Note that since we can only bend our neck so far, the avatar's body only appears at the

very edge of the image (figure 4). 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.

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.

Appendix I - User Input and Navigation

- *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 are effectively blind to the outside world. 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 of where their mouse is before they put 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 intuitive and simple the controller, the more comfortable a user will be when using it without visual reference.

Given the constraints of VR and the Rift, 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 games let users aim the camera with their head; for example, the user aims a reticle or cursor that is centered in whatever direction they're currently facing, and uses other input devices for the rest of their controls. Further research is necessary to determine the effectiveness and comfort of such a control scheme, but is already implemented in several early VR games.

The Rift sensors use information on orientation and acceleration (and in the future, 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 their heads/bodies forward to move in that direction.

We warn intrepid innovators to perform extensive user testing before unleashing a new method of control on the public; numerous unintentionally nauseating and/or frustrating prototypes have passed through the Oculus VR offices already. Head tilt is a particularly dangerous movement to experiment with, as tilting the head off of the body's axis during virtual rotation can create a "pseudo coriolis effect,"[23] which researchers have found induces motion sickness in test subjects. When done properly, however, new input methods have the potential to be more comfortable and intuitive than traditional devices.

Navigation

For most users, locomotion will be carried out 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 engaged in research into 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.

---

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

For now, traditional input methods are a safe 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 awake 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 (e.g., a jump in perspective 30° to the right, moving them to another location in the same map) can be disorienting and—if camera control is taken away to achieve the change in location—uncomfortable.

Appendix J - Content Creation

- *Keep in mind that users can and should be able to look in any direction at any time; doing so should not break immersion*
- *Beware of limitations in pixel density when creating detailed art assets*
- *Low-polygon "cheats" (like bump mapping or flat objects) can become glaringly obvious in stereoscopic 3D, particularly up close*
- *Sound is critical to immersion; design soundscapes carefully and consider the output devices users will use.*
- *Oculus tools operate in meters; treat 1 unit in Unity as 1 meter*
- *For an ideal experience, use the Oculus config tool settings for setting the user's size in the virtual environment (optional)*

Novel Demands

Designing virtual worlds can be demanding. The designer has much less direct control over where a user is going to look, since the user can turn and look anywhere at any time. As stated previously, limiting the camera's response to head movement can be very uncomfortable in VR. Constraining the camera's range of movement for narrative or technical purposes is therefore impossible. Be sure that looking around at any time does not break the user's sense of immersion—for example, by revealing any technical cheats in rendering the environment. The virtual world should always be complete and continuous around the user.

Try to engage as many dynamic systems as possible, including physics, lighting, weather, and decay. Remember that stereoscopic effects work best on objects that are fairly close; as they move farther away, the effects flatten out.

Art Assets

Although future Rift devices will have improved resolution, their wider field of view means it will be a while before they can match the pixel density of a conventional display. Even though the Rift has limited resolution, it is capable of creating a fully immersive experience—as long as your content is mindful of this limitation. As the scale of objects approaches the size of a single row of pixels, detailed rendering will become problematic. The thinner an object is, the worse the clarity when viewed on the Rift. Fine detail—such as text or small, thin objects—will have a tendency to get lost between the pixels. Similarly, objects made up of thin repeating elements, such as fences and patterns, can be problematic to display.

When creating your worlds, make sure to view them on the Rift at all stages of the design process. Be on the lookout for objects that seem to flicker in and out of existence. Avoid extremely small objects; if possible, avoid making thin objects. These recommendations apply to textures just as much as they do to geometry.

Most real-time 3D applications, like games, use a number of techniques that allow them to render complex scenes at acceptable frame rates. Some effects that effectively accomplish that goal look obviously fake in stereoscopic 3D. Billboard sprites can look

very obviously flat, particularly if they have sharp detail on them (e.g. lightning, fire). Try to only use billboards for hazy objects, such as smoke or mist, or distant background elements. Bumpmapping is not very effective in VR unless combined with parallax mapping or real geometry (in this case, make sure to do the parallax mapping correctly for each virtual eye).

At the current limited resolution of the Oculus Rift, you can still get away with many of the same tricks used in non-VR games, but as resolution improves, these tricks will become more obvious to the user.

Audio Design

Audio is one of the two principal modalities used in virtual worlds. High-quality audio can make up for lower-quality visual experiences, and since audio resources are usually less processor-intensive than visual resources, putting emphasis on audio can be a useful strategy.

Sound is arguably as important to immersion as vision. Hearing sounds from your surroundings continuously supports your perception of those surroundings. To make a virtual environment believable, the user needs to be convinced that the part of the world that is currently out of view still exists; sound can serve as a powerful cue that the world is still there.

A natural complement to the Oculus Rift is a pair of headphones, and many users will likely wear them during gameplay. In VR, headphones and speakers have different demands for spatializing audio. Virtual microphones need to act as the user's ears in the virtual environment the same way virtual cameras act as the user's eyes. Audio design should take into account the fact that when wearing headphones, the audio output sources follow the user's ears with any head movement. The same is not true of speaker systems.

The virtual "microphones" that capture environmental audio should always follow the user's *position*; however, their behavior is different between headphones and speakers. When the user is wearing headphones, the microphones need to rotate in *orientation* with the user's head (based on the Rift's head orientation sensors). When using speakers, head movement should not affect microphone orientation. Your content should support both modes, with an option for the user to select either speakers or headphones.

To take audio design further, true 3D spatialization can be created using head-related transfer functions (HRTF). Many sound libraries already support HRTFs (including Miles, DirectSound and OpenAL), and developers should use them.

User and Environment Scale

Scale is a critical aspect of VR. Just as in the physical world, users have a visceral sense of the size of objects in relation to their own body, and it's easy to tell when any object

(or the entire world) is set to the wrong scale. For most games, you'll want to make sure to get everything scaled correctly. The Oculus Rift software, which handles inter-camera distance and field of view, expects everything to be measured in meters, so you'll want to use the meter as your reference unit.

There are three degrees of freedom to the user's physical size in VR: the height of the user's eyes from the ground, the size of camera movements in response to head motions, and the distance between the user's pupils (the IPD). All of these are supplied by the SDK or the user's profile, but the application may wish to manipulate them in various ways. By default, we advise not altering them—using the user's real-world dimensions will maximize comfort and immersion while preventing disorientation.

In many games, the avatar the user inhabits may need to be a specific height, either for the purposes of narrative or gameplay. For example, you may wish for certain parts of the environment to reliably block users' view, or they might need to be a certain height to properly interact with elements in the environment. Fixing the user at a height in the virtual world that does not match their own should not create problems, as long as it does not cause a moving ground plane to fill their field of view (which intensifies the experience of vection). However, you must continue to preserve the user's IPD and head motions for their comfort.

It is possible to achieve some Alice in Wonderland style effects by scaling the whole world up or down. Since scale is relative, there are actually two ways to make your user feel enormous or tiny: you could scale the world, or you could scale the user (via height and ICD). For now, we advise scaling the world around the user to avoid potential issues with simulator sickness that can arise from scaling the user in the virtual environment.

If you're working in Unity, treat a single unit as a meter to most closely approximate real-world dimensions. If you are creating a real-world VR experience, you'll be set. If you are creating a fantasy or alternate reality VR experience, you may consider a different sense of scale. Whether or not you are aiming for realism, we recommend testing out your art assets inside the Rift early and throughout the design process. Make sure your world scale balances a comfortable experience while achieving your intended artistic effect.

As a caveat, we should note that the research literature has found that people's perception tends to underestimate distance in virtual environments,[24][25] so even perfectly measured worlds can still seem a little off at times.

---

[24] Messing, R. & Durgin, F.H. (2005). Distance perception and the visual horizon in head-mounted displays. *ACM Transcriptions on Applied Perception, 2(3)*, 234-250.

[25] Willemsen, P., Colton, M. B., Creem-Regehr, S. H., & Thompson, W. B. (2004, August). The effects of head-mounted display mechanics on distance judgments in virtual environments. In *Proceedings of the 1st Symposium on Applied Perception in Graphics and Visualization* (pp. 35-38). ACM.

Appendix K - Closing thoughts on effective VR (for now)

- *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 over-reach 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 designing of a 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 http://developer.oculusvr.com/best-practices for the latest information and discussions on designing VR content for the Rift.

Appendix L - Health and Safety Warnings

Please read the warnings below carefully before using the Headset to reduce the risk of personal injury, sickness or property damage.

Immediately Discontinue Use: if anyone using the Headset has any of the following symptoms: convulsions; seizures; eye or muscle twitching; involuntary movements, dizziness; disorientation; altered vision; loss of awareness; nausea; lightheadedness; simulation sickness (similar to motion sickness); or any type of discomfort or pain in the head or eyes.

General Instructions & Precautions: You should always follow these instructions and observe these precautions while using the Headset to reduce the risk of injury or sickness:

- The Headset should be calibrated for each individual user by using the calibration software before putting on the Headset and starting a virtual reality experience. Failure to follow this instruction may increase the risk of discomfort and simulation sickness.
- Read and follow all setup and operating instructions provided with the Headset.
- See a doctor before using the Headset if you are pregnant, elderly, or suffer from a heart condition or other serious medical condition.
- See a doctor if you have serious symptoms, or symptoms that continue after stopping use of the Headset.
- Do not drive, operate machinery or engage in similar activities until you have recovered from any symptoms you experienced.
- Ease into the use of the Headset to allow your body to adjust, beginning with only a few minutes at a time.
- Take a 10 to 15 minute break every hour, even if you don't think you need it. Each person is different, so take more frequent and longer breaks if you feel discomfort.
- Avoid excessive usage of the Headset.
- If your eyes become tired or sore or if you feel dizzy or nauseated while using the Headset, stop and rest.
- Use the Headset in a seated position whenever possible.
- Do not use the Headset when you are tired, need sleep, or are under the influence of alcohol or drugs.

Seizures: Some people (about 1 in 4000) may have severe dizziness, seizures, epileptic seizures or blackouts triggered by light flashes or patterns, and this may occur while they are watching TV, playing video games or experiencing virtual reality, even if they have never had a seizure or blackout before or have no history of seizures or epilepsy. Anyone who has had a seizure, loss of awareness, or other symptom linked to an epileptic condition should see a doctor before using the Headset.

Simulation Sickness, Eye Strain, & Other Reactions: Content available for use on the Headset produces an immersive virtual reality experience, and users may have reactions to that experience, including simulation sickness (similar to motion sickness), perceptual after-effects, disorientation, decreased postural stability, eye strain, dizziness and/or nausea, and feelings of depersonalization (feelings of watching yourself act) or derealization (feelings that the external world seems unreal). These reactions may be triggered when using the Headset for a brief or sustained period of time. See a doctor before using the Headset if you or anyone in your family has a history of experiencing these symptoms.

Children:

- This product should not be used by children under the age of 7. Viewing of 3D images by children under age 7 may cause permanent vision damage.
- Parents should watch their children (age 7 and older) who are using the Headset for any of the symptoms described above.
- Parents should monitor their children (age 7 and older) for appropriate time using the Headset.

Awareness of Surroundings: Content available for use on the Headset produces an immersive virtual reality experience that distracts you from your actual surroundings. Always be aware of your surroundings when using the Headset. Remain seated whenever possible, and take special care to ensure that you are not near other people, stairs, balconies, windows, furniture, or other items that you can run into, trip over, or knock down when using, or immediately after using, the Headset, and do not handle sharp or otherwise dangerous objects while using the Headset.

Radio Frequency Interference: The Headset can emit radio waves that can affect the operation of nearby electronics, including cardiac pacemakers.

- Do not operate the Headset within 9 inches of a pacemaker while using the wireless feature.
- If you have a pacemaker or other implanted medical device, do not use the wireless feature of the Headset without first consulting your doctor or the manufacturer of your medical device.
- Observe and follow all regulations and rules regarding use of wireless devices in locations such as hospitals, airports, and on board aircraft. Operation in those locations may interfere with or cause malfunctions of equipment, with resulting injuries to persons or damage to property.

Electrical Shock Hazard: To reduce the risk of electric shock:

- Do not expose the Headset to water or moisture.
- Unplug the Headset before cleaning, and clean only with a dry cloth.
- Keep the Headset away from open flames and other heat sources.
- Do not modify or disassemble the Headset.