

Figure 21.7 **Mock crime scenes captured with a 3rdTech laser scanner. Even though artifacts such as cracks and shadows can be seen in parts of the scene that the laser scanner could not see (e.g., behind the victim's legs), the results can be quite compelling.** (Image courtesy of 3rdTech)

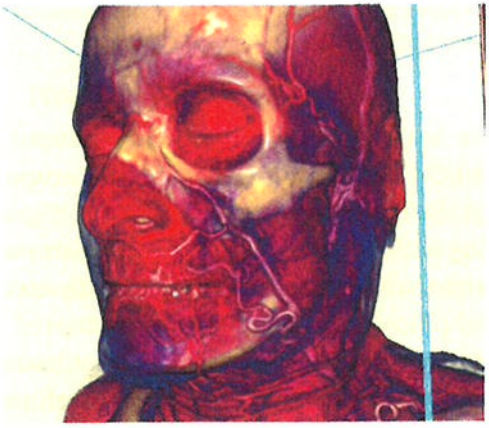


Figure 21.8 **A real-time voxel-based visualization of a medical dataset. The system enables the user to explore the dataset by "crawling" through it with a 3D multi-touch interface.** (Image courtesy of Digital ArtForms)

Visualizing data from within VR can provide significant insight for scientists to understand their data. Multiple depth cues such as head-motion parallax and real walking that are not available with traditional displays enable scientists to gain insight by better seeing shape, protrusions, and relationships they previously did not know existed even when they were, or so they thought, intimately knowledgeable about their data.

Meta Platforms, Inc.
Exhibit 1067
Page 270



Figure 21.9 **Whereas this dataset of a fibrin network (green) grown over platelets (blue) may seem like random polygon soup to those not familiar with the subject, scientists intimately familiar with the dataset gained significant insight after physically walking through the dataset with an HMD. Even with such simple rendering, the wide range of depth cues and interactivity provides understanding not possible with traditional tools.** (Courtesy of UNC CISMM NIH Resource 5-P41-EB002025 from data collected in Alisa S. Wolberg's laboratory under NIH award HL094740.)

From March 2008 to March 2010, scientists from various departments and organizations used the UNC-Chapel Hill Department of Computer Science VR Lab to physically walk through their own datasets (Taylor, 2010) (Figure 21.9). In addition to clarifying understanding of their datasets, several scientists were able to gain insight that was not possible with traditional displays they typically used. They understood the complexity of structural protrusions better; saw concentrated material next to small pieces of yeast that led to ideas for new experiments; saw branching protrusions and more complex structures that were completely unexpected; more easily tracked and navigated branching structures of lungs, nasal passages, and fibers; and more easily saw clot heterogeneity to get an idea of overall branch density. In one case, a scientist stated, "We didn't do the experiment right to answer the question about clumping . . . we didn't know that these tools existed to view the data this way." If they were able to view the data in an HMD before the experiment, then the experiment design could have been corrected.

Meta Platforms, Inc.
Exhibit 1067
Page 271

Affecting Behavior

VR users who feel fully present inside the creator's world can be affected by content more than in any other medium. This chapter discusses several ways that the designer can influence and affect users through mechanisms linked to content, as well as how the designer should consider hardware and networking when creating content.

22.1 Personal Wayfinding Aids

22.1.1 Maps

A **map** is a symbolic representation of a space, where relationships between objects, areas, and themes are conveyed. The purpose of a map is not necessarily to provide a direct mapping to some other space. Abstract representations can often be more effective in portraying understanding. Consider, for example, subway maps that only show relevant information and where scale might not be consistent.

Maps can be static or dynamic and looked at before navigation or concurrently while navigating. Maps that are used concurrently involve placement of oneself on the map (mentally or through technology) and help to answer the questions "Where am I?" and "What direction am I facing?"

Dynamic "you-are-here" markers on a map are extremely helpful for users to understand where they are in the world. Showing the user's direction is also important. This can be done with an arrow or the user's field of view on the map.

For large environments, it can be difficult to see details if the map represents the entire environment. The scale of the environment can be modified, such as by a pinch gesture or 3D Multi-Touch Pattern (Section 28.3.3). To prevent the map from getting in the way/being annoying when it is not of use, consider putting the map on the non-dominant hand and give the option to turn it on/off.

Maps can also be used to more efficiently navigate to a location by entering the map through the World-In-Miniature Pattern (Section 28.5.2)—point on the map where to go and then "enter into" the map with the map becoming the surrounding world.

Meta Platforms, Inc.
Exhibit 1067
Page 272

How a map is oriented can have a big impact on spatial understanding. Map use during navigation is different from map use for spatial knowledge extraction. While driving or walking, people in an unfamiliar area generally turn maps in different directions, and misaligned maps cause problems for some people in judgments of direction. However, when looking over a map to plan a trip, the map is normally not turned. Depending on the task, maps are best used with either a forward-up orientation or a north-up orientation.

Forward-up maps align the information on the map to match the direction the user is facing or traveling. For example, if the user is traveling southeast, then the top center of the map corresponds to that southeast direction. This method is best used when concurrently navigating and/or searching in an egocentric manner (Darken and Cevik, 1999), as it automatically lines up the map with the direction of travel and enables easy matching between points on the map and corresponding landmarks in the environment. When the system does not automatically align the map (i.e., the users have to manually align the map), many users will stop using the map.

North-up maps are independent of the user's orientation; no matter how the user turns or travels, the information on the map does not rotate For example, if the user holds the map in the southeast direction, then the top center of the map still represents north. North-up maps are good for exocentric tasks. Example exocentric uses of north-up maps are (1) familiarizing oneself with the entire layout of the environment independent of where one is located, and (2) route planning before navigating. For egocentric tasks, a mental transformation is required from the egocentric perspective to the exocentric perspective.

22.1.2 Compasses

A compass is an aid that helps give a user a sense of exocentric direction. In the real world, many people have an intuitive feel of which way is north after only being in a new location for a short period of time. If virtual turns are employed in VR, then keeping track of which way is which is much more difficult due to the lack of physical turning cues. By having easy access to a virtual compass, users can easily redirect themselves after performing a virtual turn and return to the intended direction of travel. Compasses are similar to forward-up maps except compasses only consist of direction, not location.

A virtual compass can be held in the hand as in the real world. Holding the compass in the hand enables one to precisely determine the heading of a landmark by holding the compass up to directly overlay ticks on the compass onto landmarks in the environment. In contrast to the real world, a virtual compass can hang in space relative to the body (Section 26.3.3) so there is no need to hold it with the hand, although

Meta Platforms, Inc.
Exhibit 1067
Page 273

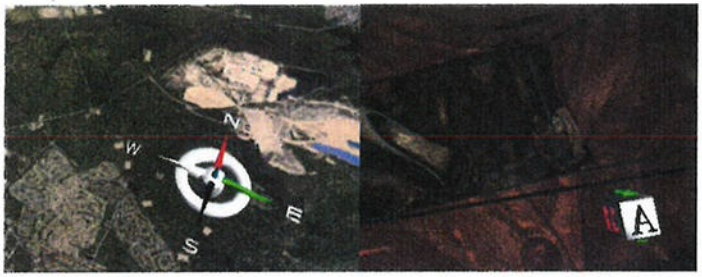


Figure 22.1 **A traditional cardinal compass (left) and a medical compass with a human CT dataset (right). The letter *A* on the medical compass represents the anterior (front) side of the body, and the letter *R* represents the right lateral side of the body.** (Images courtesy of Digital ArtForms)

this comes at the disadvantage of not being able to easily line up the compass with landmarks. The hand, however, can easily grab and release the compass as needed. Figure 22.1 (left) shows a standard compass showing the cardinal directions over a terrain.

A compass does not necessarily provide cardinal directional information. Dataset visualization sometimes uses a cubic compass that represents direction relative to the dataset. For example, medical visualization sometimes uses a cube with the first letter of the directions on each face of the cube to help users understand how the close-up anatomy relates to the context of the entire body. Figure 22.1 (right) shows such a compass.

If the compass is placed at eye level surrounding the user, then the user can overlay the compass ticks on landmarks by simply rotating the head (in yaw and pitch). Alternatively, the compass can be placed at the user's feet as shown in Figure 22.2. In this case the user simply looks down to determine the way he is facing. This has the advantage of not being distracting as the compass is only seen when looking in the downward direction.

22.1.3 Multiple Wayfinding Aids

Individual wayfinding aids by themselves are not particularly useful. Combining multiple aids together can significantly help. Effectiveness of wayfinding depends on the number and quality of wayfinding cues or aids provided to users (Bowman et al., 2004). However, too many wayfinding aids can be overwhelming; choose what is most appropriate for the goals of the project.

Meta Platforms, Inc.
Exhibit 1067
Page 274

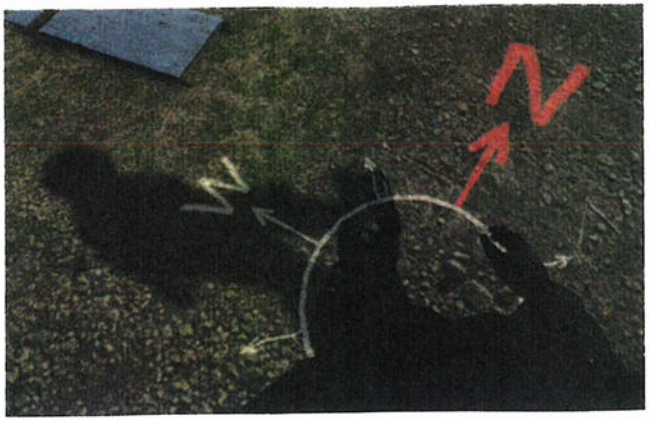


Figure 22.2 **A compass surrounding the user's feet.** (Image courtesy of NextGen Interactions)

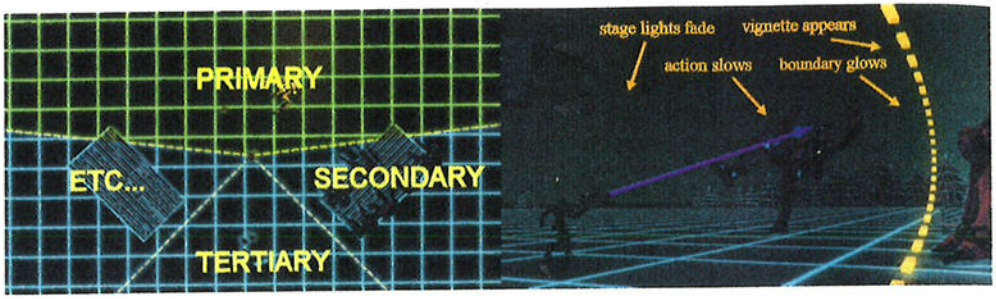


Figure 22.3 **Visionary VR splits the scene into different action zones.**

22.2 Center of Action

Not being able to control what direction users are looking is a new challenge for filmmakers as well as other content makers. Visionary VR, a start-up based in Los Angeles, splits the world around the user into different zones such as primary and secondary viewing directions (Figure 22.3) (Lang, 2015). The most important action occurs in the primary direction. As a user starts looking toward another zone, cues are provided to inform the user that she is starting to look at a different part of the scene and the action and/or content is about to change if they keep looking in that direction. For example, glowing boundaries between zones can be drawn, the lights might dim for a zone as the user starts to look toward a different zone, or time might slow down to a halt as users look toward a different zone. Zones are also aware if the user is looking in their direction so actions within that zone can react when being

Meta Platforms, Inc.
Exhibit 1067
Page 275

looked at. These techniques can be used to enable users to experience all parts of the scene independent of the order in which they view the different zones.

22.3 Field of View

There are a number of techniques designers can use to take advantage of the power of field of view (Mark Bolas, personal communication, June 13, 2015). One is to artificially constrain the field of view for certain parts of a VR experience, and then allow it to go wide for others. This is similar to how cinematographers use light or color in a movie; they control the perceptual arc to correlate with moments in the story being experienced. Another is to shape the frame around the field of view—rounded, asymmetric shapes feel better than rectilinear ones. You want the periphery to feel natural, even including the occlusion due to the nose. The qualitative result is the ability to author content with greater range and emotional depth. Its power turns up in unexpected ways. For example, anecdotal evidence suggests virtual characters cause a greater sense of presence when looking at them with a wide field of view compared to a narrow field of view. When there is a wide field of view, there is more of an obligation to obey social norms—for example, not staring at characters for too long and keeping an appropriate distance.

22.4 Casual vs. High-End VR

Different types of systems each have their own advantages and disadvantages. The experiences will be very different depending on the system type, and the experience should be targeted and designed for the primary type (although multiple types might be supported, the experience should be most optimized for one). This section focuses on wired vs. wireless and mobile vs. location-based systems. Chapter 27 discusses system details specific to user input.

22.4.1 Wired vs. Wireless VR

Wired and wireless systems result in different experiences whether the designer considers the trade-offs or not. For example, users will feel the tug of wires on the HMD resulting in a break-in-presence if they attempt to physically turn too far with a wired system. To optimize the user experience, design should take into account whether the targeted system is wired or wireless.

Wired seated systems should place content in front of the user and/or use an interface that enables virtual turning so that wires do not become tangled. An advantage of not enabling virtual turning (in addition to minimizing motion sickness) is that the designer can assume the user is facing in a general forward direction. Many seated

Meta Platforms, Inc.
Exhibit 1067
Page 276

experiences can be designed so that non-tracked hand controllers can be assumed to be held in the lap, such that some visual representation of the controller and user's hands can be drawn at the approximate location of the physical hands (Section 27.2.2). If virtual turning is used, the visual hands/controller should turn with the virtual turn (e.g., draw some form of chair-like device with the controller and body attached to the chair).

Freely turnable systems are wireless systems that allow users to easily look in any direction just as they would in the real world without concern of which direction is forward in the real world. Real turning has the advantages that it does not cause motion sickness and does not cause disorientation. A system that includes a swivel chair or omnidirectional treadmill should use a wireless system as cables will otherwise quickly become tangled after more than a 180° turn.

Fully walkable systems enable users to stand up and physically walk around. Fully walkable untethered systems are known as **nomadic VR** (Denise Quesnel, personal communication, June 8, 2015). Whereas fully walkable systems can work extremely well, a non-immersed spotter should follow behind the user to prevent injury, and in the case of wired systems to prevent entanglement, tugging of the cables on the user's head, and breaks-in-presence by the wires touching any part of the user's body. Even with a wireless system or wires attached to a rail system, such as that at UNC-Chapel Hill, a spotter should still be used.

22.4.2 Mobile vs. Location-Based VR

Mobile VR vs. location-based VR is the equivalent of mobile apps/games vs. more detailed and higher-powered desktop software. Although there is overlap, each should be designed to be optimized to take advantage of its specific characteristics (similar to input devices).

Mobile VR is defined by being able to place all equipment in a small bag and being able to immerse oneself almost instantaneously at about any time and any location when engagement with the real world is not required. Cell phones that snap into a head-mounted device are currently the standard for mobile VR. Mobile VR is also much more social and casual as VR experiences can be shared on a plane, at a party, or in a meeting.

Location-based VR requires a suitcase or more, takes time to set up, and ranges from one's living room or office to out-of-home experiences (Williams and Mascioni, 2014) such as VR arcades or theme parks. Location-based VR has the potential to be higher quality and the most immersive as it can include high-end equipment and tracking technologies.

Meta Platforms, Inc.
Exhibit 1067
Page 277



Figure 22.4 **A user drives a virtual car in a CAVE while speaking with an avatar controlled by a remote user.** (From Daily et al. (2000))

22.5 Characters, Avatars, and Social Networking

A character can be either a computer-controlled character or an avatar. An **avatar** is a character that is a virtual representation of a real user. This section discusses characters and how they influence behavior. Section 32.3 discusses technical implementation of social networking. Figure 22.4 shows an example of a user and a remote user rendered locally as an avatar.

As mentioned in Section 4.3, how we perceive ourselves can be quite distorted. In VR, we can explicitly define and design ourselves from a third-person perspective to look any way we wish. In addition, audio filters and prerecorded audio can be used to modify the sound of our voice. Providing the capability for users to personalize their own avatar has been proven to be very popular in both 2D virtual worlds (e.g., Second Life) as well as more fully immersive worlds. The Xbox Avatar store alone has 20,000 items for free or purchase.[1] **Caricature** is a representation of a person or thing in which certain striking characteristics are exaggerated and less important features are omitted or simplified in order to create a comic or grotesque effect. Cartoonlike rendering with caricature can be enjoyable, effective, compelling, and presence inducing. Caricature can be especially effective for avatars because it can avoid the Uncanny Valley (Section 4.4.1).

1. http://marketplace.xbox.com/en-US/AvatarMarketplace; accessed June 16, 2015.

Meta Platforms, Inc.
Exhibit 1067
Page 278



Figure 22.5 **Robotic characters without legs prevent breaks-in-presence that often occur when characters walk or run due to the difficulty of animating walking/running in a way that humans perceive as real.** (Image courtesy of NextGen Interactions)

As discussed in Section 9.3.9, the human ability to perceive the motion of others is extremely good. Although motion capture can result in character animation that is extremely compelling, this is difficult to do when mixing motion capture data with characters that are able to arbitrarily move and turn at different speeds. A simple solution is to use characters without legs, such as legless robotic characters, so that no breaks-in-presence occur due to walking and animations not always looking correct. Figure 22.5 shows an example of an enemy robot that floats above the ground and only tilts when moving. Although not as impressive as a human-like character with legs, simple is better in this case to prevent breaks-in-presence resulting from awkward-looking leg movement.

Character head motion is extremely important for maintaining a sense of social presence (Pausch et al., 1996). Fortunately, since the use of HMDs requires head tracking, this information can be directly mapped to one's avatar as seen by others (assuming quality bandwidth). Eye animations can add effect but are not necessary as any three-year-old child can attest to after watching *Sesame Street*. Moving an avatar's eyes that does not result from eye tracking also risks the false impression that the character is not paying attention.

Meta Platforms, Inc.
Exhibit 1067
Page 279



Figure 22.6 **An example of a fully immersive VR social experience.** (Image courtesy of VRChat)

To naturally convey attention, computer-controlled characters should first turn their heads before turning their bodies. Rotating the eyes before head turning can also add effect. However, if a computer-controlled character has eye motion and a user-controlled avatar does not, then that can provide a cue to users of which other characters are user controlled and which are not. Characters can also be used to encourage users to look in certain directions by pointing or lining themselves up between the user and the object to be looked at.

Social networking across distances through technology is not new. Like online 2D virtual worlds, people now spend hours at a time in fully immersive social VR such as VRChat (Figure 22.6). Jesse Joudrey (personal communication, April 20, 2015), CEO of VRChat, claims the urge to communicate with others in VR to be much stronger than in other digital mediums. For example, there is an urge to back up in VR if someone gets in one's personal space. Although he believes face-to-face interaction is the best way of communicating, VR is the second-best way to interact with real people. Interestingly, even though users can visually define themselves differently than what they look like in real life, behavior is more difficult to change. Involuntary body language is difficult to hide even with only basic motion. People who are fidgety in real life also fidget in VR.

Meta Platforms, Inc.
Exhibit 1067
Page 280

Transitioning to VR Content Creation

It is interesting to realize that while the VR industry is rushing to develop VR game systems, the real task will be to fill these systems with interesting and entertaining environments. This task is challenging because the medium is much richer and demanding than existing video games. While a conventional game requires the designer to create a two-dimensional video environment, the nature of VR requires that a completely interactive and rich environment be created.

—Mark Bolas (1992)

VR development is very different than traditional product or software development. This chapter provides some of the most important points to consider when creating VR content.

23.1 Paradigm Shifts from Traditional Development to VR Development

VR is very different than any other form of media. In order to fully understand and take advantage of VR, it is very useful to approach it as a new artistic medium for expression (Bolas, 1992). VR is extremely cross-disciplinary and thus it is useful to study and understand other disciplines. However, no single discipline is sufficient by itself. There will be things that work from other disciplines and some that won't. Be prepared to give up on those things that don't work. The following are some key points that should always be at the forefront of a VR content creator's mind.

Focus on the user experience. For VR, the user experience is more important than for any other medium. A bad website design is still usable. A bad VR design will make users sick. The experience must be extremely compelling and engaging to convince people to stick a piece of hardware on their face.

Minimize sickness-inducing effects. Whereas real-world motion sickness is common (e.g., seasickness or from a carnival ride), sickness is almost never an issue

Meta Platforms, Inc.
Exhibit 1067
Page 281

with other forms of digital media. Whereas more traditional digital media does occasionally cause sickness (e.g., IMAX and 3D film), it is minor compared to what sickness can occur with VR if not designed properly (Part III).

Make aesthetics secondary. Aesthetics are nice to have, but other challenges are more important to focus on at the beginning of a project. Start with basic content and incrementally build upon it while the essential requirements are maintained. For example, frame rate is absolutely essential in reducing latency, so geometry should be optimized as necessary. In fact, frame rate is so important that a core requirement of all VR systems should be to match or exceed the refresh rate of the HMD, and the scene should fade out when this requirement is not met (Section 31.15.3).

Study human perception. Perceptions are very different in VR than they are on a regular display. Film and video games can be viewed from different angles with almost no impact on the experience. VR scenes are rendered from a very specific viewpoint of where the user is looking from. Depth cues are important for presence (Section 9.1.3), and inconsistent sensory cues can cause sickness (Section 12.3.1). Understanding how we perceive the real world is directly relevant to creating immersive content.

Give up on having all the action in one part of the scene. No other form of media completely surrounds and encompasses the user. The user can look in any direction at any time. Many traditional first-person video games allow the user to look around with a controller, but there is still the option for the system to control the camera when necessary. Proper VR does not offer such luxury. Provide cues to direct users to important areas but do not depend upon them.

Experiment excessively. VR best practices are not yet standardized (and they may never be) nor are they mature—there are a lot of unknowns so there must be room for experimentation. What works in one situation might not work in another situation. Test across a variety of people that fit within your target audience and make improvements based on their feedback. Then iterate. Iterate. Iterate . . . (Part VI).

23.2 Reusing Existing Content

Traditional digital media is very different from VR. With that being said, some content can be included or even ported to VR. 2D images or video is very straightforward to apply onto a texture map within a virtual environment. In fact, watching 2D movies within VR has proven to be surprisingly popular. Web content or any other content

Meta Platforms, Inc.
Exhibit 1067
Page 282

can even be updated in real time and interacted with on a 2D surface such as that described in Section 28.4.1.

Although there are similarities, video game design is very different than VR design, primarily due to being designed for a 2D screen and not having head tracking. Concepts such as head tracking, hand tracking, and the accurate calibration required to match how we perceive the real world are core elements for VR experiences that the designer should consider from the start of the project. If any of these essential VR elements are not implemented well, then the mind will quickly reject any sense of presence and sickness may result. Because of these issues, porting existing games to VR without a redesign and rewrite of the game engine where appropriate is not recommended. Reusing assets and refactoring code where appropriate is the correct solution. With that being said, implementing the right solution requires resources and there will be those who port existing games with no or minimal modification of code. Thus, the following information is provided to help port games for those choosing this path.

23.2.1 Geometric Detail

Games typically use texture excessively in place of 3D geometry. More depth cues, especially stereo cues, and motion parallax cues make textures obviously 2D. Whereas such flat cardboard-appearing worlds are not necessarily realistic, there is no concern for flat textures to cause sickness.

In addition, many standard video game techniques and geometric hacks simplify 3D calculations for better performance. Simplified geometry and lighting created with screen space techniques, normal maps, textures, billboard sprites, some shaders and shadows (depending on implementation), and other illusion of 3D in games are not actually 3D and often look flat or strange in VR. Many of these techniques fortunately often do work at further distances but should be avoided for close distances. For close objects to be most convincing, they should be modeled in detail. Such detailed objects can be swapped out with lower levels of detailed models and with geometric hacks when they become further from the viewpoint. Some code may need to be modified or rewritten to appear correct in VR, even for further objects.

23.2.2 Heads-Up Displays

Heads-up displays (HUDs) are especially problematic for VR. This is because most HUDs are implemented as a 2D overlay that is in screen space, not part of the scene. Because there is no depth, most ports simply place the same HUD image to the left and right eyes, resulting in binocular cues being at an infinite depth. Unfortunately, this results in a binocular-occlusion conflict (Section 13.2), which is a major problem.

Meta Platforms, Inc.
Exhibit 1067
Page 283

When the two eyes see the same image of the HUD, the HUD appears at an infinite distance so the entire scene should occlude the HUD, but it does not. Some solutions for this are described below.

1. The easiest solution is to turn off all HUD elements. Some game engines enable the user to do this without modifying source code. However, some of the HUD information is important for status and game play, so this solution can leave players quite confused.
2. Draw or render the HUD information to a texture and place onto a transparent quad at some depth in front of the user (and possibly angled and below eye level so the information is accessed by looking down) so that it becomes occluded when other geometry passes in front of it. Factors such as readability, occlusion of the scene, and accommodation-vergence conflict can occur, but these are relatively minor compared to binocular-occlusion conflict and adjustments can be made until reasonably comfortable values are found. This solution works reasonably well, except that the HUD information becomes occluded when other geometry passes in front of it.
3. Separate out the different HUD elements and place them on the user's body and/or in different areas relative to the user's body (e.g., an information belt or at the user's feet). This is the ideal solution but can take substantial work depending on the game and game engine.

23.2.3 Targeting Reticle

Targeting reticles are typically implemented as part of the HUD. However, the solutions for a VR reticle may be different.

1. The reticle can be implemented in the same way as option #2 for HUDs as described above. This results in the reticle behaving as a helmet with the reticle on the front of the helmet in front of the eyes. Like a HUD, a challenge with this solution is the reticle is seen as a double image when looking at the target in the distance. This is due to the eyes not being converged on the reticle and is congruent with how one sights a weapon in the real world. This is a more realistic solution as the real world requires one to close one eye to properly aim a weapon. One eye (ideally the system should enable the user to configure the reticle for his dominant eye) must be set to be the sighting eye so that the reticle crosshairs directly overlap the target. In order to know what the user is targeting, the system must know what eye is being used in order to cast a ray from the eye through the reticle.

Meta Platforms, Inc.
Exhibit 1067
Page 284

2. Another option is to project the reticle onto (or just in front of) whatever is being aimed at. Although not as realistic and the reticle jumps large distance in depth, this is preferred by some users as being more comfortable and is less confusing when one doesn't understand the need to close one eye to aim.

23.2.4 Hand/Weapon Models

Similar to HUDs, the hands and/or weapons in desktop games are placed directly in front of the user, either as 2D textures or full 3D objects. The obvious problem is that 2D textures representing the hands/weapons will appear flat and not work well with VR.

Although not realistic and perhaps a bit disturbing at first, replacing 2D hands and arms with 3D hands without arms that are directly controlled with a tracked hand-held controller can work quite well in VR. However, hands with arms are more of a problem. In most first-person desktop video games there is no need to have a full body, so when the user looks down in a game ported to VR there is no body, or at best only some portion of the body. If the game includes hands, then the arms will likely be hanging in space. Rotating a tracked hand-held controller cannot be directly mapped to an entire static hand/arm model because the point of rotation should be about the hand, and rotation of the hand results in the entire arm rotating. In addition, many game engines have a completely different implementation for the hands and weapons than the rest of the world, so generalizing the hands and weapons to behave as other 3D objects may not be possible without re-architecting or refactoring the code. If arms are required, then some form of inverse kinematics will be required.

23.2.5 Zoom Mode

Some games have a zoom mode with a tool or weapon such as a sniper rifle. A zoom lens that moves with the head causes a difference between the physical field of view and the rendered field of view, which results in an unstable virtual world and sickness when the head moves (a similar result can occur with real-world magnifying glasses; Section 10.1.3). Because of this, the zoom should be independent of head pose or only a small portion of the screen should be zoomed in (i.e., a small part of the screen where the scope is located). If the zoom is affected by head motion and it is not possible to prevent a large part of the scene from zooming in, then zooming should be disabled.

Meta Platforms, Inc.
Exhibit 1067
Page 285

Content Creation: Design Guidelines

Virtual worlds begin as an entirely open slate for the content designer to build upon. The preceding four chapters and the following guidelines will help to lay the foundation and paint the details of newly created worlds.

High-Level Concepts of Content Creation (Chapter 20)

Experiencing the Story (Section 20.1)

- Focus on creating and refining a great experience that is enjoyable, challenging, and rewarding.
- Focus on conveying the key points of the story instead of the details of everything. The users should all consistently get those essential points. For the non-essential points, their minds will fill in the gaps with their own story.
- Use real-world metaphors to teach users how to interact and move through the world.
- To create a powerful story, focus on strong emotions, deep engagement, massive stimulation, and an escape from reality.
- Focus on the experience instead of the technology.
- Provide a relatable background story before immersing users into the virtual world.
- Make the story simple and clear.
- Focus on why users are there and what there is to do.
- Provide a specific goal to perform.
- Focus on believability instead of photorealism.
- Keep the experience tame enough for the most sensitive users.

Meta Platforms, Inc.
Exhibit 1067
Page 286

- Minimize breaks-in-presence.
- Focus on the targeted audience.

The Core Experience (Section 20.2)

- Keep the core experience the same even with an array of various contexts and constraints.
- Make the core experience enjoyable, challenging, and rewarding so users want to come back for more.
- If the core experience is not implemented well, then no number of fancy features will keep users engaged for more than a few minutes.
- Continuously improve the core experience, build various prototypes focused on that core, and learn from observing and collecting data from real users.

Conceptual Integrity (Section 20.3)

- The core conceptual model should be consistent.
- Make the basic structure of the world directly evident and self-explanatory.
- Eliminate extraneous content and features that are immaterial to the intended experience.
- Have one good idea rather than several loosely related good ideas. If there are too many good incompatible ideas, then reconsider the primary idea to provide an overarching theme that encompasses all of the best ideas in a coherent manner.
- Empower a director who controls the basic concept to lead the project.
- The director defines the project, but is not a dictator of implementation or data collection.
- The director should always give full credit, even when others have implemented his ideas.
- The director should be fully accessible to the makers and data collectors, and highly encourage questions.

Gestalt Perceptual Organization (Section 20.4)

- Use gestalt principles of grouping when creating assets and user interfaces.
- Use gestalt principles of grouping at a higher conceptual level, such as keeping objects simple and temporal closure.
- Use concepts of segregation—ground acts as a stable reference and figure emphasizes objects that can be interacted with.

Meta Platforms, Inc.
Exhibit 1067
Page 287

Environmental Design (Chapter 21)

The Scene (Section 21.1)

- Simplify the background and contextual geometry. Focus on fundamental geometry and interactive objects.
- Make sure geometry scaling is consistent. For realistic experiences, include familiar objects with standard sizes that are easily seen by the user.

Color and Lighting (Section 21.2)

- Use color to bring out emotions.
- Use multiple colors and only slight variations of white light so users perceive the intended color of objects and color constancy is maintained.
- Capture users' attention with bright colors that stand out.
- Turn on and off color on specific objects to signify use or disable use.

Audio (Section 21.3)

- Use audio for situational awareness from all directions, adding emotional impact, cuing visual attention, conveying information without taxing the visual system, and providing unique cues that cannot be perceived through other sensory systems.
- Do not make people deaf by not providing audio.
- Do not overwhelm with sound.
- Use aggressive warning sounds only occasionally to grab attention.
- Use ambient sound effects to provide a sense of realism and presence.
- Use music to evoke emotion.
- Use audio with interfaces as feedback to the user.
- Use audio along with color to replace touch when haptics are not available.
- Convey information through the spoken word.

Sampling and Aliasing (Section 21.4)

- Avoid spatially high-frequency visual elements.

Environmental Wayfinding Aids (Section 21.5)

- Use environmental wayfinding aids to help users maintain a sense of their position and direction of travel, know where goals are located, and plan in their minds how to get there.

Meta Platforms, Inc.
Exhibit 1067
Page 288

- Use both explicit and subtle wayfinding aids.
- Study the work of non-VR designers such as architectural design and urban planners.
- Place landmarks strategically.
- Provide tall landmarks so they can be seen from most places in the environment.
- Differentiate areas of the environment with regions that have different characteristics.
- Use channels to restrict navigation options while providing a feeling of openness.
- Place signs at nodes.
- Use edges to deter travel.
- Use "handrails" to guide navigation.
- Utilize simple spatial organization such as grid-like patterns.
- Minimize violations of the overarching metaphor of the world structure.
- Adding structure to abstract or scientific data can provide context, but be careful of imposing inappropriate structure on data that does not exist in the data itself. Consult with subject-matter experts before adding such cues to datasets.
- Implement breadcrumbs, trails, markup tools, user-placed markers, and measurement tools for users to better understand and think about the environment, as well as to share information with other users.

Real-World Data (Section 21.6)

- For immersive film, reconsider the entire movie experience as the viewer being within and a part of the scene.
- When capturing the world in 360°, make sure that all equipment and people that are not part of the story are not visible to the camera.
- For 360° camera capture, remove stereoscopic cues in the down and up directions.
- Capture light fields or true 3D data to enable motion parallax.
- Empower scientists by enabling them to walk through their own datasets and see that data in new ways.

Meta Platforms, Inc.
Exhibit 1067
Page 289

Affecting Behavior (Chapter 22)

Personal Wayfinding Aids (Section 22.1)

- Use abstract representations in maps to portray understanding of essential environmental concepts.
- Use "you are here" maps with arrows or field-of-view markings to convey both location and direction.
- For large environments, allow the map to be scaled.
- Use forward-up maps when concurrently navigating and/or searching in an egocentric manner in order to enable easy matching between points on the map and corresponding landmarks in the environment.
- Use north-up maps for exocentric tasks (e.g., when users want (1) to familiarize themselves with the entire layout of the environment independent of where they are located or (2) to plan routes before navigating).
- Use compasses so users can easily direct themselves in the direction of intended travel.
- Place a compass around the user, such as floating in space where it can be easily grabbed, surrounding the user at eye level, or surrounding the user at the feet.

Center of Action (Section 22.2)

- Consider breaking the scene around the user into zones.

Casual vs. High-End VR (Section 22.4)

- When creating content, take into account whether the system is wired or wireless.
- If virtual turning is implemented, move the controls/hands with the torso.
- For wired systems, put the majority of content in the general forward direction.
- Design and optimize for a single system, even when supporting multiple types of systems.

Characters, Avatars, and Social Networking (Section 22.5)

- When the bandwidth is available, map head tracking data to avatars' heads.
- Unless the intent is for users to distinguish between computer-controlled character behavior and user-controlled avatars, make the computer-controlled characters behave in a similar manner as avatars (e.g., similar head movements).

Meta Platforms, Inc.
Exhibit 1067
Page 290

- Be careful of adding artificial eye movements to avatars to avoid the risk of conveying that the remote user is not paying attention.
- To convey natural attention of a computer-controlled character, first move the eyes (if implemented), then the head, then the body.
- To get users to look in certain directions, use computer-controlled characters to place themselves between the user and the intended look direction, or have the characters look and/or point in the intended direction.
- For non-realistic experiences, use caricature to exaggerate the most important features of characters.
- Consider not using legs for characters that move in order to reduce breaks-in-presence caused by awkward-looking leg movement.
- Provide the ability for users to personalize their own avatars.

Transitioning to VR Content Creation (Chapter 23)

Paradigm Shifts from Traditional Development to VR Development (Section 23.1)

- Study human perception and other disciplines, but be prepared to give up on concepts that do not work with VR.
- Focus on the user experience as the experience is more important for VR than for any other medium.
- Minimize sickness-inducing effects.
- Calibrate often and excessively. Poor calibration is a major cause of motion sickness.
- Make aesthetics secondary. Start instead with basic content and incrementally build while focusing on ease of use and maintaining essential requirements.
- Give up on having all the action in one part of the scene.
- Experiment excessively. There are unknowns and various forms of VR—what works in one situation might not work in another situation.
- Consider core concepts of VR such as head and hand tracking from the start of the project.

Reusing Existing Content (Section 23.2)

- Instead of directly porting desktop applications to VR, reuse assets and refactor code where needed.

Meta Platforms, Inc.
Exhibit 1067
Page 291

- Understand that geometric hacks (e.g., representing depth with textures) that work well with desktop systems often appear strange in VR.
- Be very careful of using desktop heads-up displays and targeting reticles as straightforward porting/implementation causes binocular-occlusion conflicts.
- Replace desktop 2D hands/weapons with 3D models.
- When using tracked hand-held controllers, do not render the arms unless inverse kinematics is available.
- Be careful of using a zoom mode.

Meta Platforms, Inc.
Exhibit 1067
Page 292

PART

INTERACTION

For this book, interaction is defined to be a subset of general communication (Section 1.2). **Interaction** is the communication that occurs between a user and the VR application that is mediated through the use of input and output devices. An **interface** is the VR system side of the interaction—it exists whether a user is interacting or not.

Interaction in VR at first seems obvious—simply interact with the virtual world in a similar way that we interact with the real world. Unfortunately, natural real-world interfaces (and Hollywood interfaces) often do not work well within VR as one would expect. This is not only because virtual worlds do not model the real world as closely as we would like, but even if they did, abstract interfaces are often superior (e.g., if one wanted to look up information in a virtual world, one would not want to have to travel to a virtual library to find a virtual book). Whereas some of the basic concepts from traditional desktop interfaces can be used within immersive environments, there are few similarities between the two. Creating quality interactions is currently one of the greatest challenges for VR.

Well-implemented interaction techniques enable high levels of performance and comfort while diminishing the impact of human and hardware limitations. It is the role of the interaction designer to make complex interactions intuitive and effective.

Part V consists of five chapters that provide an overview of the most important aspects of VR interaction design.

Chapter 25, Human-Centered Interaction, reviews core concepts for general human-centered interaction: intuitiveness, Norman's principles of interaction design, direct vs. indirect interaction, the cycle of interaction, and the human hands.

Meta Platforms, Inc.
Exhibit 1067
Page 293

Chapter 26, VR Interaction, discusses some of the most important distinctions especially relevant to VR interaction, such as interaction fidelity, proprioceptive and egocentric interaction, reference frames, multimodal interaction, and unique challenges of VR interaction.

Chapter 27, Input Devices, provides an overview of input device characteristics and classes of various input devices. No single input device is appropriate for all VR applications, and choosing the most appropriate input device for the intended experience is essential for optimizing interactions.

Chapter 28, Interaction Patterns and Techniques, describes several patterns and examples of various interaction techniques. An interaction pattern is a generalized high-level interaction concept that can be used over and over again across different applications to achieve common user goals. An interaction technique is a more specific implementation of an interaction pattern.

Chapter 29, Interaction: Design Guidelines, summarizes the previous four chapters and lists a number of actionable guidelines for developing interaction techniques that largely depend on application needs.

Meta Platforms, Inc.
Exhibit 1067
Page 294

Human-Centered Interaction

VR has the potential to provide experiences and deliver results that cannot be otherwise achieved. However, VR interaction is not just about an interface for the user to reach their goals. It is also about users working in an intuitive manner that is a pleasurable experience and devoid of frustration. Although VR systems and applications are incredibly complex, it is up to designers to take on the challenge of having the VR application effectively communicate to users how the virtual world and its tools work so that those users can achieve their goals in an elegant manner.

Perhaps the most important part of VR interaction is the person doing the interacting. Human-centered interaction design focuses on the human side of communication between user and machine—the interface from the user's point of view. Quality interactions enhance user understanding of what has just occurred, what is happening, what can be done, and how to do it. In the best case, not only will goals and needs be efficiently achieved, but the experiences will be engaging and enjoyable.

This section describes several general human-centered design concepts that are essential for interaction designers to consider when designing VR interactions.

25.1 Intuitiveness

Mental models (Section 7.8) of how a virtual world works are almost always a simplified version of some complexity. When interacting, there is no need for users to understand the underlying algorithms—just the high-level relationship between objects, actions, and outcomes. Whether a VR interface attempts to be realistic or not, it should be intuitive. An **intuitive interface** is an interface that can be quickly understood, accurately predicted, and easily used. Intuitiveness is in the mind of the user, but the designer can help form this intuitiveness by conveying through the world and interface itself concepts that support the creation of a mental model.

Meta Platforms, Inc.
Exhibit 1067
Page 295

An **interaction metaphor** is an interaction concept that exploits specific knowledge that users already have of other domains. Interaction metaphors help users quickly develop a mental model of how an interaction works. For example, VR users typically think of themselves as "walking" through an environment, even though in most implementations they are not physically moving their feet (e.g., they may be controlling the walking with a hand-held controller). For such an interface, users assume they are traveling at a set height above a surface, and if that height changes, then the interaction does not fit the mental model and the user will likely become confused.

Mental models can be made more consistent across users by providing manuals, communicating with other users, and from the virtual world itself. For fully immersive VR, users are cut off from the real world and there is no guarantee they will look at a manual or talk with others. Thus, the virtual world should be sufficient in projecting appropriate information needed to create a consistent conceptual model of how things work in the mind of each user, without requiring external explanation. Otherwise, problems will occur when an application does not fit with the user's mental model. It should not be assumed that an expert will always be available to directly explain how an interface works, answer questions, or correct mistakes. Too often, a VR creator expects the user's model to be identical to what she thinks she has designed, but unfortunately this is rarely the case since creators and users think differently (as they should). Tutorials that cannot be completed until the user has indicated a clear understanding and effective interaction are a great method of inducing mental models into the minds of users.

25.2 Norman's Principles of Interaction Design

When users interact in VR, they need to figure out how to work the system. **Discoverability** is exploring what something does, how it works, and what operations are possible (Norman, 2013). Discoverability is especially important for fully immersive VR because the user is blind and deaf to the real world, cut off from those real-world humans who want to help. Essential tools can lead the mind into discovering how an interface works through consistent affordances, unambiguous signifiers, constraints to guide actions and ease interpretation, immediate and useful feedback, and obvious and understandable mappings. These principles as defined by Norman are summarized below along with how they relate to VR and are a good starting point when designing and refining interactions.

25.2.1 Affordances

Affordances define what actions are possible and how something can be interacted with by a user. We are used to thinking that properties are associated with objects, but

Meta Platforms, Inc.
Exhibit 1067
Page 296

an affordance is not a property; an affordance is a *relationship* between the capabilities of a user and the properties of a thing. Interface elements afford interaction, such as a virtual hand affording selection. An affordance between an object and one user may be different between that object and another user. Light switches on a wall offer the ability to control the lighting in a room, but only for those who are able to reach the switches. Some objects in virtual environments afford selecting, moving, controlling, etc. Good interaction design focuses on creating appropriate affordances to make desired actions easily doable with the technology used (e.g., a tracking system that is able to track the hand near the point of the light switch) and by the intended user base (e.g., the designer may intentionally place a light switch to not be reachable by a certain class of users in order to encourage collaboration).

25.2.2 Signifiers

Some affordances are perceivable, others are not. To be effective, an affordance should be perceivable. A **signifier** is any perceivable indicator (a signal) that communicates appropriate purpose, structure, operation, and behavior of an object to a user. A good signifier informs a user what is possible before she interacts with its corresponding affordance.

Examples of signifiers are signs, labels, or images placed in the environment indicating what is to be acted upon, which direction to gesture, or where to navigate toward. Other signifiers directly represent an affordance—such as the handle on a door or the visual and/or physical feel of a button on a controller. A misleading signifier can be ambiguous or not represent an affordance—something may look like a drawer to be opened when in fact it cannot be opened. Such a false signifier is usually accidental or not yet implemented. But a misleading signifier can also be purposeful—such as to motivate users to find a key in order to turn the non-accessible drawer into something that can be opened. In such cases, the content creator should be aware of such anti-signifiers and be careful not to frustrate users.

Signifiers are most often intentional, but, as mentioned above, they may also be accidental. An example of an intentional signifier is a sign giving directions. In the real world, an example of an accidental and unintentional (but useful) signifier is garbage on a beach representing unhealthy conditions. At first thought, we might think signifiers are only intentionally created in VR, for the VR creator created everything from that which does not actually exist. However, this is not always the case. An unintended VR signifier might be an object that looks like it is designed to be picked up to be placed into a puzzle, but it can also be perceived as an object that can be picked up and thrown (a common occurrence much to the frustration of content creators). Or an unintended signifier in a social VR experience might be a gathering of users at an

Meta Platforms, Inc.
Exhibit 1067
Page 297

area of interest, signifying to others to navigate to that location to investigate what is happening and what affordance might be available there.

Signifiers might not be attached to a specific object. Signifiers can be general information. Conveying what interaction mode a user is currently using can help prevent confusion.

Regardless of how and why signifiers are created, signifiers are important for communicating to users whether or not action is possible, and what those actions are. Good VR design ensures signifiers are effectively discoverable through signifiers that are well communicated and intelligible.

25.2.3 Constraints

Interaction constraints are limitations of actions and behaviors. Such constraints include logical, semantic, and cultural limitations to guide actions and ease interpretation. With that being said, this section focuses on the physical and mathematical constraints as such constraints most directly apply to VR interactions. For an overview of general project constraints, see Section 31.10.

Proper use of constraints can limit possible actions, which makes interaction design feasible and can simplify interaction while improving accuracy, precision, and user efficiency (Bowman et al., 2004). A commonly used method for simplifying VR interactions is to constrain interfaces to only work in a limited number of dimensions. The **degrees of freedom** (DoF) for an object are the number of independent dimensions available for the motion of that object (also see Section 27.1.2). The DoF of an interface can be constrained by the physical limitations of an input device (e.g., a physical dial has one DoF, a joystick has two DoF), the possible motion of a virtual object (e.g., a slider on a panel is constrained to move along a single axis to control a single value), or travel that is constrained to the ground, enabling one to more easily navigate through a scene.

Physical limitations constrain possible actions and these limitations can also be simulated in software. In addition to being useful, constraints can also add more realism. For example, a virtual hand can be stopped even when there is nothing physically stopping a user's real hand from going through the object (although this results in visual-physical conflict; Section 26.8). However, physics should not always necessarily be used as physics can sometimes make interactions more difficult. For example, it can be useful to leave virtual tools hanging in the air when not using them.

Interaction constraints are more effective and useful if appropriate signifiers make them easy to perceive and interpret, so users can plan appropriately before taking action. Without effective signifiers, users might effectively be constrained because they are not able to determine what actions are possible. Consistency of constraints

Meta Platforms, Inc.
Exhibit 1067
Page 298

can also be useful as learning can be transferred across tasks. People are highly resistant to change; if a new way of doing things is only slightly better than the old, then it is better to be consistent. For experts, providing the ability to remove constraints can be useful in some situations. For example, advanced flying techniques might be enabled after users have proved they can efficiently maneuver by being constrained to the ground.

25.2.4 Feedback

Feedback communicates to the user the results of an action or the status of a task, helps to aid understanding of the state of the thing being interacted with, and helps to drive future action. In VR, timely feedback is essential. Something as simple as moving the head requires immediate visual feedback, otherwise the illusion of a stable world is broken and a break-in-presence results (and, even worse, motion sickness). Input devices capture users' physical motion that is then transformed into visual, auditory, and haptic feedback. At the same time, feedback internal to the body is generated from within—proprioceptive feedback that enables one to feel the position and motion of the limbs and body. Unfortunately, it is quite difficult for VR to provide all possible types of feedback. Haptic feedback is especially difficult to implement in a way similar to how real forces occur in the real world. Section 26.8 discusses how sensory substitution can be used in place of strong haptic cues. For example, objects can make a sound, become highlighted, or vibrate a hand-held controller when a hand collides with them or they have been selected.

Feedback is essential for interaction, but not when it gets in the way of interaction. Too much feedback can overwhelm the senses, resulting in cluttered perception and understanding. Feedback should be prioritized so less important information is presented in an unobtrusive manner and essential information always captures attention. Instead of putting information on a heads-up display in the head reference frame, place it near the waist or toward the ground in the torso reference frame so the information is easily accessed when needed but not in the way otherwise. If information must always be visible with a heads-up display, then only provide the most essential minimal information as anything directly in front of a user reduces awareness of the virtual world. Too many audio beeps or, worse, overlapping audio announcements, can cause users to ignore all of them or even cause them to be indecipherable even if users wanted to listen to them. Such clutter not only can result in unusable application, but can be annoying (think of backseat drivers!) and is inappropriate. In many cases, users should have the option to turn off or turn down feedback that is not important to them.

Meta Platforms, Inc.
Exhibit 1067
Page 299

25.2.5 Mappings

A **mapping** is a relationship between two or more things. The relationship between a control and its results is easiest to learn where there is an obvious understandable mapping between controls, action, and the intended result. Mappings are useful even when the person is not directly holding the thing being manipulated, for example, when using a screwdriver to lift a lever up that cannot be directly reached.

Mappings from hardware to interaction techniques defined by software are especially important for VR. Often a device will have a natural mapping to one technique but poor mapping to another technique. For example, hand-tracked devices work well for a virtual hand technique and for pointing (Section 28.1) but not as well for a driving simulator where a physical steering wheel would be more appropriate.

Compliance

Compliance is the matching of sensory feedback with input devices across time and space. Maintaining compliance improves user performance and satisfaction. Compliance results in perceptual binding (Section 7.2.1) so that interaction feels as if one is interacting with a single coherent object. Visual-vestibular compliance is especially important for reducing motion sickness (Section 12.3.1). Compliance can be divided into spatial compliance and temporal compliance (Bowman et al., 2004) as described below.

Spatial compliance. Direct spatial mappings lead immediately to understanding. For example, we intuitively understand that once we have grabbed an object, to move the object up, we simply move the hand holding the object up. **Spatial compliance** consists of position compliance, directional compliance, and nulling compliance.

Position compliance is the co-location of sensory feedback with the input device position. An example of spatial compliance is when the proprioceptive sense of where the hand is matches the visual sense of where the hand is. Not all interaction techniques require position compliance, but position compliance results in direct intuitive interaction and should be used whenever appropriate. Labels placed at the location where physical controls are located on a hand-held device is an example of position compliance. Position compliance is important for tracked physical devices that the user can pick up and/or interact with. Consider users who have just placed an HMD on their head but have not yet picked up the hand controllers (or who have previously set the controller down); the user must be able to see the controller in the correct location in order to pick it up.

Directional compliance is the most important of the three spatial compliances. **Directional compliance** states virtual objects should move and rotate in the same

Meta Platforms, Inc.
Exhibit 1067
Page 300

direction as the manipulated input device. This results in correspondence of what is seen and what is felt by the body, resulting in more direct interaction. This enables the user to effectively anticipate motion in response to physical input and therefore plan and execute appropriately.

An example of directional compliance is the mapping from a mouse to a cursor on the screen. Even though a mouse is spatially dislocated from the screen (i.e., it lacks position compliance), the hand/mouse movement results in an immediate and isomorphic movement of the cursor on the screen, which makes the user feel as if he is directly moving the cursor itself (Veer and Melguizo, 2002). The user does not think about the offset between the mouse and the cursor when the mouse is used as intended. Even though the mouse typically sits on a flat horizontal surface and the screen sits vertically (i.e., the screen is rotated 90° from the mouse), intuitively we expect if we move the mouse forward/back then the cursor will move up/down in the same way because we think of both as up/down movements. However, for anyone who has attempted to use a mouse when it is rotated by 90° to the right, they know simple manipulations can be extremely difficult due to up becoming left and right becoming up. The same is true for moving objects in VR; the direction of movement for a grabbed object, even if at a distance, should be mapped directly to that selected object whenever possible. Likewise, when a user rotates a virtual object with an input device, the virtual object should rotate in the same direction; that is, both should rotate around the same axis of rotation (Poupyrev et al., 2000).

Nulling compliance states that when a device returns to its initial placement, the corresponding virtual object should also return to its initial placement (Buxton, 1986). Nulling compliance can be accomplished with absolute devices (relative to some reference), but not relative devices (relative to itself)—see Section 27.1.3. For example, if a device is attached to the user's belt, nulling compliance is important, as the user can use "muscle memory" to remember the initial, neutral placement of the device and corresponding virtual object (Section 26.2).

Temporal compliance. Temporal compliance states that different sensory feedback corresponding to the same action or event should be synced appropriately in time. Viewpoint feedback should be immediate to match vestibular cues, otherwise motion sickness may result (Chapter 15). But even for reasons unrelated to sickness, feedback should be immediate; otherwise users may become frustrated or give up before tasks are completed. Even if the entire action cannot be completed immediately, there should be some form of feedback implying the problem is being worked on. Without such information, users can become annoyed and computing resources can be wasted since the user may have forgotten about the task and moved on to something else. In

Meta Platforms, Inc.
Exhibit 1067
Page 301

fact, slow or poor feedback may be worse than no feedback, as it can be distracting, irritating, and anxiety provoking. Anyone who has attempted to browse the Web on an extremely slow Internet connection can attest to this.

Non-spatial Mappings

Non-spatial mappings are functions that transform a spatial input into a non-spatial output or a non-spatial input into a spatial output. Some indirect spatial to non-spatial mappings are universal, such as moving the hand up signifies more and moving the hand down signifies less. Other mappings are personal, cultural, or task dependent. For example, some people think of time moving from left to right whereas others think of time moving from behind the body to the front of it.

25.3 Direct vs. Indirect Interaction

Different types of interaction can be thought of as being on a continuum from indirect interaction to direct interaction. Both direct and indirect interaction is useful for VR depending upon the task. Use each where appropriate and do not use one where not appropriate, e.g., don't try to force everything to be direct.

Direct interaction is an impression of direct involvement with an object rather than of communicating with an intermediary (Hutchins et al., 1986). The most direct interaction occurs when the user directly interacts with a physical object in the hands. Well-designed hand-held tools (e.g., a knife) that directly affect an object are only slightly less direct, as once a user understands the tool, it seems to become one with the user as if the tool was an extension of the body instead of an intermediary object.

An example of direct interaction is when a user moves a virtual object on a touch screen with his finger. However, the virtual object does not follow the finger when the finger leaves the plane of the screen. VR enables more direct interactions than other digital technologies because virtual objects can be directly mapped to the hands in 3D space with full spatial (both directional and positional) and temporal compliance (Section 25.2.5). Directional compliance and temporal compliance are more important than position compliance; consider the feeling of directness with a mouse (albeit not quite as directly as moving the cursor with a finger on a touch screen), even though there is no position compliance. Similarly, manipulating an object at a distance can feel like one is directly manipulating the object.

Indirect interaction requires more cognition and conversion between input and output. Performing a search for an image through a typed or verbal query is an example of an indirect interaction. The user must put thought into what is being searched for, provide the query, wait for a response, and then interpret the result. Not only

Meta Platforms, Inc.
Exhibit 1067
Page 302

direction as the manipulated input device. This results in correspondence of what is seen and what is felt by the body, resulting in more direct interaction. This enables the user to effectively anticipate motion in response to physical input and therefore plan and execute appropriately.

An example of directional compliance is the mapping from a mouse to a cursor on the screen. Even though a mouse is spatially dislocated from the screen (i.e., it lacks position compliance), the hand/mouse movement results in an immediate and isomorphic movement of the cursor on the screen, which makes the user feel as if he is directly moving the cursor itself (Veer and Melguizo, 2002). The user does not think about the offset between the mouse and the cursor when the mouse is used as intended. Even though the mouse typically sits on a flat horizontal surface and the screen sits vertically (i.e., the screen is rotated 90° from the mouse), intuitively we expect if we move the mouse forward/back then the cursor will move up/down in the same way because we think of both as up/down movements. However, for anyone who has attempted to use a mouse when it is rotated by 90° to the right, they know simple manipulations can be extremely difficult due to up becoming left and right becoming up. The same is true for moving objects in VR; the direction of movement for a grabbed object, even if at a distance, should be mapped directly to that selected object whenever possible. Likewise, when a user rotates a virtual object with an input device, the virtual object should rotate in the same direction; that is, both should rotate around the same axis of rotation (Poupyrev et al., 2000).

Nulling compliance states that when a device returns to its initial placement, the corresponding virtual object should also return to its initial placement (Buxton, 1986). Nulling compliance can be accomplished with absolute devices (relative to some reference), but not relative devices (relative to itself)—see Section 27.1.3. For example, if a device is attached to the user's belt, nulling compliance is important, as the user can use "muscle memory" to remember the initial, neutral placement of the device and corresponding virtual object (Section 26.2).

Temporal compliance. **Temporal compliance** states that different sensory feedback corresponding to the same action or event should be synced appropriately in time. Viewpoint feedback should be immediate to match vestibular cues, otherwise motion sickness may result (Chapter 15). But even for reasons unrelated to sickness, feedback should be immediate; otherwise users may become frustrated or give up before tasks are completed. Even if the entire action cannot be completed immediately, there should be some form of feedback implying the problem is being worked on. Without such information, users can become annoyed and computing resources can be wasted since the user may have forgotten about the task and moved on to something else. In

Meta Platforms, Inc.
Exhibit 1067
Page 303

fact, slow or poor feedback may be worse than no feedback, as it can be distracting, irritating, and anxiety provoking. Anyone who has attempted to browse the Web on an extremely slow Internet connection can attest to this.

Non-spatial Mappings

Non-spatial mappings are functions that transform a spatial input into a non-spatial output or a non-spatial input into a spatial output. Some indirect spatial to non-spatial mappings are universal, such as moving the hand up signifies more and moving the hand down signifies less. Other mappings are personal, cultural, or task dependent. For example, some people think of time moving from left to right whereas others think of time moving from behind the body to the front of it.

25.3 Direct vs. Indirect Interaction

Different types of interaction can be thought of as being on a continuum from indirect interaction to direct interaction. Both direct and indirect interaction is useful for VR depending upon the task. Use each where appropriate and do not use one where not appropriate, e.g., don't try to force everything to be direct.

Direct interaction is an impression of direct involvement with an object rather than of communicating with an intermediary (Hutchins et al., 1986). The most direct interaction occurs when the user directly interacts with a physical object in the hands. Well-designed hand-held tools (e.g., a knife) that directly affect an object are only slightly less direct, as once a user understands the tool, it seems to become one with the user as if the tool was an extension of the body instead of an intermediary object.

An example of direct interaction is when a user moves a virtual object on a touch screen with his finger. However, the virtual object does not follow the finger when the finger leaves the plane of the screen. VR enables more direct interactions than other digital technologies because virtual objects can be directly mapped to the hands in 3D space with full spatial (both directional and positional) and temporal compliance (Section 25.2.5). Directional compliance and temporal compliance are more important than position compliance; consider the feeling of directness with a mouse (albeit not quite as directly as moving the cursor with a finger on a touch screen), even though there is no position compliance. Similarly, manipulating an object at a distance can feel like one is directly manipulating the object.

Indirect interaction requires more cognition and conversion between input and output. Performing a search for an image through a typed or verbal query is an example of an indirect interaction. The user must put thought into what is being searched for, provide the query, wait for a response, and then interpret the result. Not only

Meta Platforms, Inc.
Exhibit 1067
Page 304

must the query be thought about but the conversion from words to and from visual imagery must be considered. Even though such indirect interaction requires more cognition than direct interactions, that does not mean direct interaction is always better. Indirect interactions are more effective for their intended purposes.

Somewhere in the middle of the extremes of direct and indirect interactions are semi-direct interactions. An example is a slider on a panel that controls the intensity of a light. The user is directly controlling the intermediary slider but less directly controlling the light. However, once the slider is grabbed and moved back and forth a couple of times it feels more direct due to the mental mapping of up for lighter and down for darker.

25.4 The Cycle of Interaction

Interaction can be broken down into three parts: (1) forming the goal, (2) executing the action, and (3) evaluating the results (Norman, 2013).

Execution bridges the gap between the goal and result. This feedforward is accomplished through the appropriate use of signifiers, constraints, mappings, and the user's mental model. There are three stages of execution that follow from the goal: plan, specify, and perform.

Evaluation enables judgment about achieving a goal, adjustments, or the creation of new goals. This feedback is obtained by perceiving the impact of the action. Evaluation consists of three stages: perceive, interpret, and compare.

As can be seen in Figure 25.1, the seven stages of interaction consist of one stage for goals, three stages for execution (plan, specify, and perform), and three stages for evaluation (perceive, interpret, and compare). Quality interaction design considers requirements, intentions, and desires at each stage. An example interaction is as follows.

1. Form the **goal.** The question is "What do I want to accomplish?" Example: Move a boulder blocking an intended route of travel.
2. **Plan** the action. Determine which of many possible plans of action to follow. Question: "What are the alternative set of action sequences and what do I choose?" Example: Navigate to the boulder or select from a distance to move it.
3. **Specify** an action sequence. Even after the plan is determined, the specific sequence of actions must be determined. Question: "What are the specific actions of the sequence?" Example: Shoot a ray from the hand, intersect the boulder, push the grab button, move the hand to the new location, release the grab button.

Meta Platforms, Inc.
Exhibit 1067
Page 305

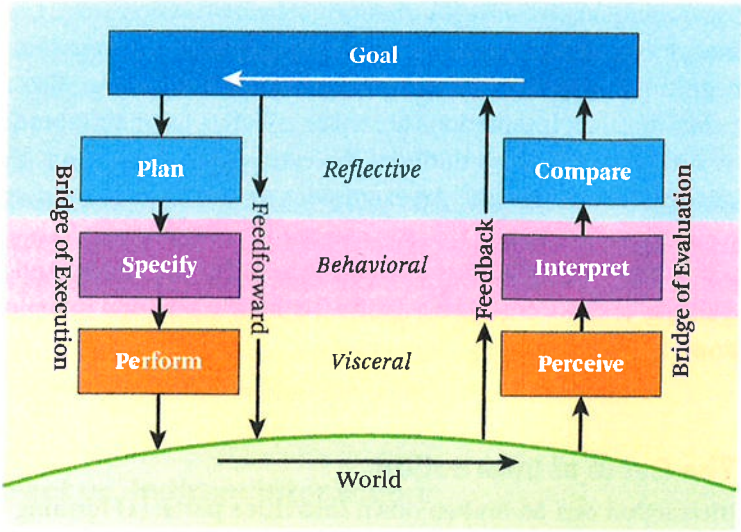


Figure 25.1 **The cycle of interaction.** (Modified from Norman (2013))

4. **Perform** the action sequence. Question: "Can I take action now?" One must actually take action to achieve a result. Example: Move the boulder.
5. **Perceive** the state of the world. Question: "What happened?" Example: The boulder is now in a new location.
6. **Interpret** the perception. Question: "What does it mean?" Example: The boulder is no longer in the path of desired travel.
7. **Compare** the outcome with the goal. Questions: "Is this okay? Have I accomplished my goal?" Example: The route has been cleared so navigation along the route is possible.

The interaction cycle can be initiated by establishing a new goal (goal-driven behavior). The cycle can also initiate from some event in the world (data-driven or event-driven behavior). When initiated from the world, goals are opportunistic rather than planned. Opportunistic interactions are less precise and certain than explicitly specified goals, but they result in less mental effort and more convenience.

Not all activities in these seven stages are conscious. Goals tend to be reflective (Section 7.7) but not always. Often, we are only vaguely aware of the execution and evaluation stages until we come across something new or run into an obstacle, at which point conscious attention is needed. At the highest reflective level of goal setting and comparison, we assess results in terms of cause and effect. The middle levels of

Meta Platforms, Inc.
Exhibit 1067
Page 306

specifying and interpreting are often only semi-conscious behavior. The visceral level of performing and perceiving is typically automatic and subconscious unless careful attention is drawn to the actual action.

Many times the goal is known, but it is not clear how to achieve it. This is known as the **gulf of execution.** Similarly, the **gulf of evaluation** occurs when there is a lack of understanding about something. Designers can help users bridge these gulfs by thinking about the seven stages of interaction, performing task analysis for creating interaction techniques (Section 32.1), and utilizing signifiers, constraints, and mappings to help users create effective mental models for interaction.

25.5 The Human Hands

The human hands are remarkable and complex input and output devices that naturally interact with other physical objects quickly, precisely, and with little conscious attention. Hand tools have been perfected over thousands of years to perform intended tasks effectively. Figure 8.8 shows that a large portion of the sensory cortex is devoted to the hands. It is not surprising, then, that the best fully interactive VR applications typically make use of the hands. Using interaction techniques that the hands can intuitively work with can add significant value to VR users.

25.5.1 Two-Handed Interaction

It is natural and intuitive to reach out with both hands to manipulate objects in the real world, so common sense says two-handed 3D interfaces should be appropriate and intuitive for VR (Schultheis et al., 2012). Whereas such intuition tells us that two hands are better than one, two hands can in fact be worse than one hand if the interaction is designed inappropriately (Kabbash et al., 1994). Perhaps this is a reason why, even though devices like the mouse have been around for decades, most computer interactions are one handed. Anyone who has developed 3D systems knows that 3D devices alone do not guarantee superior performance, and this is even more true for two hands, in part because the hands do not necessarily work in parallel (Hinckley et al., 1998). Iterative design with feedback from real users (Part VI) is essential for creating high-quality bimanual interfaces.

Bimanual Classifications

Bimanual interaction (two-handed interaction) can be classified as **symmetric** (each hand performs identical actions) or **asymmetric** (each hand performs a different action), with asymmetric tasks being more common in the real world (Guiard, 1987).

Meta Platforms, Inc.
Exhibit 1067
Page 307

Bimanual symmetric interactions can further be classified as **synchronous** (e.g., pushing on a large object with both hands) or **asynchronous** (e.g., climbing a ladder with one arm reaching up at a time). Scaling by grabbing two sides of an object and spreading the hands apart is an example of a bimanual symmetric interaction.

Bimanual asymmetric interactions occur when both hands work differently but in a coordinated way to accomplish a task. The non-dominant hand provides the reference frame, giving the ergonomic benefit of placing the object being worked upon in a way that is comfortable (often subconsciously) for the dominant hand to work with and that does not force the dominant hand to work in a single locked position. The non-dominant hand also typically initiates manipulation of the task and performs gross manipulation of the object being manipulated, in order to provide convenient, efficient, and precise manipulation by the dominant hand. The most commonly given example is writing; the non-dominant hand controls the orientation of the paper for writing by the dominant hand. Or consider peeling a potato—the potato is much easier to peel when held in the non-dominant hand than when sitting on a table (whether the potato is locked in place or not). In a similar manner, unimanual interactions in VR can be awkward when the non-dominant hand does not control the reference frame for the dominant hand.

By using two hands in a natural manner, the user can specify spatial relationships, not just absolute positions in space. Bimanual interactions ideally should be designed so the two hands work together in a fluid manner, switching between symmetric and asymmetric modes depending on the current task.

Meta Platforms, Inc.
Exhibit 1067
Page 308

VR Interaction Concepts

VR interactions are not without their challenges. Trade-offs must be considered that may result in interactions being different than in the real world. However, there are enormous advantages VR has over the real world as well. This chapter focuses upon interaction concepts, challenges, and benefits specific to VR.

26.1 Interaction Fidelity

VR interactions are designed on a continuum ranging from an attempt to imitate reality as closely as possible to in no way resembling the real world. Which goal to strive toward depends on the application goals, and most interactions fall somewhere in the middle. **Interaction fidelity** is the degree to which physical actions used for a virtual task correspond to the physical actions used in the equivalent real-world task (Bowman et al., 2012).

On the high end of the interaction fidelity spectrum are **realistic interactions**—VR interactions that work as closely as possible to the way we interact in the real world. Realistic interactions strive to provide the highest level of interaction fidelity possible given the hardware being used. Holding one hand above the other as if holding a bat and swinging them together to hit a virtual baseball has high interaction fidelity. Realistic interactions are often important for training applications, so that which is learned in VR can be transferred to the real-world task. Realistic interactions can also be important for simulations, surgical applications, therapy, and human-factors evaluations. If interactions are not realistic in such applications, problems such as adaptation (Section 10.2) may occur, which can lead to negative training effects for the real-world task being trained for. An advantage of using realistic interactions is that there is little learning required of users since they already know how to perform the actions.

On the other end of the interaction fidelity spectrum are **non-realistic interactions** that in no way relate to reality. Pushing a button on a non-tracked controller to shoot a laser from the eyes is an example of an interaction technique that has low interaction

Meta Platforms, Inc.
Exhibit 1067
Page 309

fidelity. Low interaction fidelity is not necessarily a disadvantage as it can increase performance, cause less fatigue, and increase enjoyment.

Somewhere in the middle of the interaction fidelity spectrum are **magical interactions** where users make natural physical movements but the technique makes users more powerful by giving them new and enhanced abilities or intelligent guidance (Bowman et al., 2012). Such magical hyper-natural interactions attempt to create "better" ways of interacting by enhancing usability and performance through superhuman capabilities and non-realistic interactions (Smith, 1987). Although not realistic, magical often uses interaction metaphors (Section 25.1) to help users quickly develop a mental model of how an interaction works. Consider interaction metaphors as a source of inspiration for creating new magical interaction techniques. Grabbing an object at a distance, pointing to fly through a scene, or shooting fireballs from the hand are examples of magical interactions. Magic interactions strive to enhance the user experience by reducing interaction fidelity and circumventing the limitations of the real world. Magic works well for games and teaching of abstract concepts.

Interaction fidelity is a multi-dimensional continuum of components. The Framework for Interaction Fidelity Analysis (McMahan, 2015) categorizes interaction fidelity into three concepts: biomechanical symmetry, input veracity, and control symmetry.

Biomechanical symmetry is the degree to which physical body movements for a virtual interaction correspond to the body movements of the equivalent real-world interaction. Biomechanical symmetry makes heavy use of posture and gestures that replicate how one positions and moves their body in the real world. This provides a high sense of proprioception resulting in a high sense of presence since the user feels his body physically acting in the environment as if he were performing the task in the real world. Real walking for VR navigation purposely has a biomechanical symmetry to how we walk in the real world. Walking in place has a lower biomechanical symmetry due to its less realistic movements. Pressing a button or joystick to walk forward has no biomechanical symmetry.

Input veracity is the degree to which an input device captures and measures information. Three aspects that dictate the quality of input veracity are accuracy, precision, and latency. A system with low input veracity can significantly affect performance due to difficulty in controlling input.

Control symmetry is the degree of control a user has for an interaction as compared to the equivalent real-world interaction. High-fidelity techniques provide the same control as the real world without the need for different modes of interaction. Low control symmetry can result in frustration due to the need to switch between techniques to obtain full control. For example, directly manipulating object position and rotations (6 DoF) with a tracked hand controller has greater control symmetry than

Meta Platforms, Inc.
Exhibit 1067
Page 310

indirectly manipulating the same object with gamepad controls because the gamepad controls (less than 6 DoF) require using multiple translation and rotational modes. However, low control symmetry can also have superior performance if implemented well. For example, non-isomorphic rotations (Section 28.2.1) can be used to increase performance by amplifying hand rotations.

26.2 Proprioceptive and Egocentric Interaction

As described in Section 8.4, proprioception is the physical sense of the position and orientation of the body and limbs. Because most VR systems do not provide a sense of touch outside of hand-held devices, proprioception can be especially important for exploiting the one real object every user has—the human body (Miné et al., 1997). The body provides an egocentric frame of reference (Section 26.3.3) in which to work, and interactions relative to the body's reference frame are more effective than techniques relying solely on visual information. In fact, eyes-off interactions can be performed in peripheral vision or even outside the field of view of the display, which reduces visual clutter. The user also has a more direct sense of control within personal space—it is easier to place an object directly with the hand rather than through less direct means.

26.2.1 Mixing Egocentric and Exocentric Interactions

Exocentric interactions consist of viewing and manipulating a virtual model of the environment from outside of it. With egocentric interaction, the user has a first-person view of the world and typically interacts from within the environment. Don't assume one or the other must be chosen. These egocentric and exocentric interactions can be mixed together so that the user can view himself on a smaller map (Sections 22.1.1 and 28.5.2), and/or manipulate the world in an exocentric manner but from an egocentric perspective (Figure 26.1).

26.3 Reference Frames

A **reference frame** is a coordinate system that serves as a basis to locate and orient objects. Understanding reference frames is essential to creating usable VR interactions. This section describes the most important reference frames as they relate specifically to VR interaction. The virtual-world reference frame, real-world reference frame, and torso reference frame are all consistent with one other when there is no capability to rotate, move, or scale the body or the world (e.g., no virtual body motion, no torso tracking, and no moving the world). The reference frames diverge when that is not the case.

Although thinking abstractly about reference frames and how they relate can be difficult, reference frames are naturally perceived and more intuitively understood

Meta Platforms, Inc.
Exhibit 1067
Page 311



Figure 26.1 **An exocentric map view from an egocentric perspective.** (Image courtesy of Digital ArtForms)

when one is immersed and interacting with the different reference frames. The reference frames discussed in this section will be more intuitively understood by actually experiencing them.

26.3.1 The Virtual-World Reference Frame

The **virtual-world reference frame** matches the layout of the virtual environment and includes geographic directions (e.g., north) and global distances (e.g., meters) independent of how the user is oriented, positioned, or scaled. When creating content over a wide area, forming a cognitive map, determining global position, or planning travel on a large scale (Section 10.4.3), it is typically best to think in terms of the exocentric virtual-world reference frames. Care should be taken when placing direct hand interfaces relative to the virtual-world reference frame as reaching specific locations can be difficult and awkward unless the user is able to easily and precisely navigate and turn through the environment.

26.3.2 The Real-World Reference Frame

The **real-world reference frame** is defined by real-world physical space and is independent of any user motion (virtual or physical). For example, as a user virtually flies forward the user's physical body is still located in the real-world reference frame. A physical desk, computer screen, or keyboard sitting in front of the user is in the real-world reference frame. A consistent physical location should be provided to set

Meta Platforms, Inc.
Exhibit 1067
Page 312

any tracked or non-tracked hand-held controller when not being used. For tracked controllers or other tracked objects, make sure to match the virtual model with the physical controller in form and position/orientation in the real-world reference frame (i.e., full spatial compliance) so users can see it more correctly and more easily pick it up.

In order for virtual objects, interfaces, or rest frames to be solidly locked into the real-world reference frame, the VR system must be well calibrated and have low latency. Such interfaces often, but not always, provide output cues only to help provide a rest frame (Section 12.3.4) in order for users to feel stabilized in physical space and to reduce motion sickness. Automobile interiors, cockpits (Figure 18.1), or non-realistic stabilizing cues (Figure 18.2) are examples of cues in the real-world reference frame. In some cases it makes sense to add the capability to input information through real-world reference-framed elements (e.g., buttons located on a virtual cockpit). A big advantage of real-world reference frames is that passive haptics (Section 3.2.3) can be added to provide a sense of touch that matches visually rendered elements.

26.3.3 The Torso Reference Frame

The **torso reference frame** is defined by the body's spinal axis and the forward direction perpendicular to the torso. The torso reference frames are especially useful for interaction because of proprioception (Sections 8.4 and 26.2)—the sense of where one's arms and hands are felt relative to the body. The torso reference frame can also be useful for steering in the direction the body is facing (Section 28.3.2).

The torso reference frame is similar to the real-world reference frame in the sense that both frames move with the user through the virtual world as the user virtually translates or scales. The difference is that with the torso reference frame, virtual objects rotate with the body (both virtual and physical body turns) and move with physical translation whereas objects in the real-world reference frame do not. The chair a user is seated in can be tracked instead of the torso if it can be assumed the torso is stable relative to the chair. Systems with head tracking but not torso or chair tracking can assume the body is always facing forward (i.e., the torso reference frame and real-world reference frame are consistent). However, physical turning of the body can cause problems due to the system not knowing if only the head turned or the entire body turned.

If no hand tracking is available, the hand reference frame can be assumed to be consistent with the torso reference frame. For example, a visual representation of a non-tracked hand-held controller should move and rotate with the body (hand-held controllers are often assumed to be held in the lap). For VR, information displays often work better in torso reference frames rather than head reference frames as commonly

Meta Platforms, Inc.
Exhibit 1067
Page 313



Figure 26.2 **Information and a visual representation of a non-tracked hand-held controller in the torso reference frame.**

done with heads-up displays in traditional first-person video games. Figure 26.2 shows an example of a visual representation of a non-tracked hand-held controller and other information at waist level in the torso reference frame.

Body-Relative Tools

Just like in the real world, tools in VR can be attached to the body so that they are always within reach no matter where the user goes. This is done in VR by simply placing the tool in the torso reference frame. This not only provides convenience of the tool always being available but also takes advantage of the user's body acting as a physical mnemonic, which helps in recall and acquisition of frequently used control (Miné et al., 1997).

Items should be placed outside the forward direction so that they do not get in the way of viewing the scene (e.g., the user simply looks down to see the options and then can select via a point or grab). Advanced users should be able to turn off items or make them invisible. Examples of physical mnemonics are pull-down menus located above the head (Section 28.4.1), tools surrounding the waste as a utility belt, audio options

Meta Platforms, Inc.
Exhibit 1067
Page 314

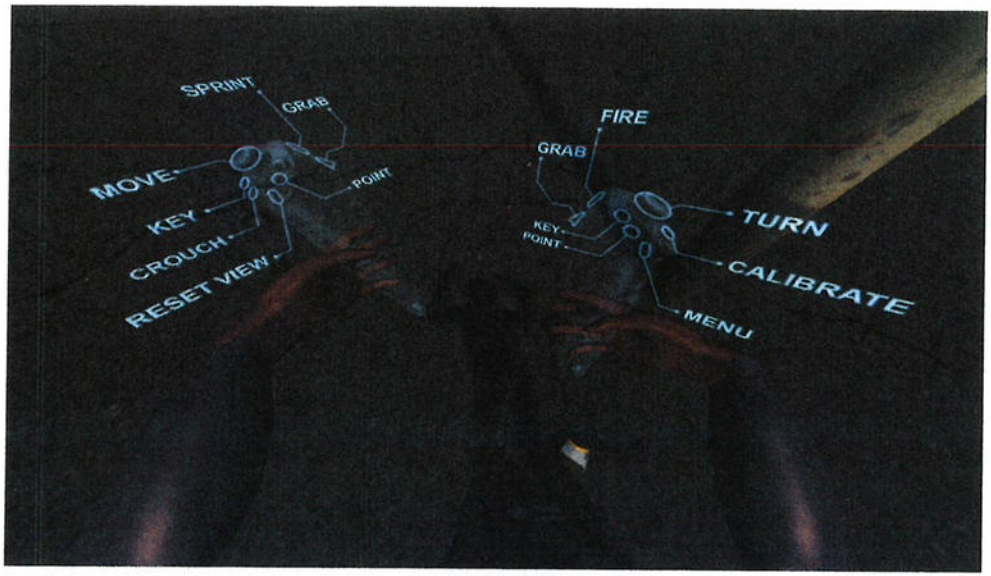


Figure 26.3 **A simple transparent texture in the hand can convey the physical interface.** (Image courtesy of NextGen Interactions)

at the ear, navigation options at the user's feet, and deletion by throwing an object behind the shoulder (and/or object retrieval by reaching behind the shoulder).

26.3.4 The Hand Reference Frames

The **hand reference frames** are defined by the position and orientation of the user's hands, and hand-centric judgments occur when holding an object in the hand. Hand-centric thinking is especially important when using a phone, tablet, or VR controller. Placing a visual representation of a tracked hand-held controller (Section 27.2.3) in the hand(s) can help add a sense of presence due to the sense of touch matching the visuals. Placing labels or icons/signifiers in the hand reference frame that point to buttons, analog sticks, or fingers is extremely helpful (Figure 26.3), especially for new users. The option to turn on/off such visuals should be provided so as to not occlude/clutter the scene when not using the interface or after the interface has been memorized. Although both the left and right hands can be thought of as separate reference frames, the non-dominant hand is useful for serving as a reference frame for the dominant hand to work in (Section 25.5.1), especially for hand-held panels (Section 28.4.1).

Meta Platforms, Inc.
Exhibit 1067
Page 315

26.3.5 The Head Reference Frame

The **head reference** frame is based on the point between the two eyes and a reference direction perpendicular to the forehead. In psychological literature, this reference frame is known as the **cyclopean eye**, which is a hypothetical position in the head that serves as our reference point for the determination of a head-centric straight-ahead (Coren et al., 1999). People generally tend to think of this straight-ahead as a direction in front of themselves, oriented around the midline of the head, regardless of where the eyes are actually looking. From an implementation point of view, the head reference frame is equivalent to the head-mounted-display reference frame, but from the user's point of view (assuming a wide field of view), the display is not visually perceived. A world-fixed secondary display that shows what the user is seeing matches the head reference frame.

Heads-up displays (HUDs) are often located in the head reference frame. Such heads-up display information should be minimized, if used at all, other than a selection pointer for gaze-directed selection (Section 28.1.2). If used, it is important to make cues small (but large enough to be easily perceived/readable), minimize the number of visual cues to not be annoying or distracting, not make the cues too far in the periphery, give the cues depth so they are occluded properly by other objects (Section 13.2), and place the cues at a far enough distance so that there is not an extreme accommodation-vergence conflict (Section 13.1). It can also be useful to make the cues transparent. Figure 26.4 shows an example HUD in the head reference frame that serves as a virtual helmet that helps the user to target objects.

26.3.6 The Eye Reference Frames

The **eye reference frames** are defined by the position and orientation of the eyeballs. Very few VR systems support the orientation portion of eye reference frames due to the requirement for eye tracking (Section 27.3.2). However, the left/right horizontal offset of the eyes is easily determined by the interpupillary distance of the specific user measured in advance of usage.

When looking at close objects (for example, when sighting down the barrel of a gun) and assuming a binocular display, eye reference frames are important to consider because they are differentiated from the head reference frame due to the offset from the forehead to the eye. The offset between the left and right eyes results in double images for close objects when looking at an object further in the distance (or double images of the further object when looking at the close object). Thus, when users are attempting to line up close objects with further objects (e.g., a targeting task), users should be advised to close the non-dominant eye and to sight with the dominant eye (Sections 9.1.1 and 23.2.2).

Meta Platforms, Inc.
Exhibit 1067
Page 316

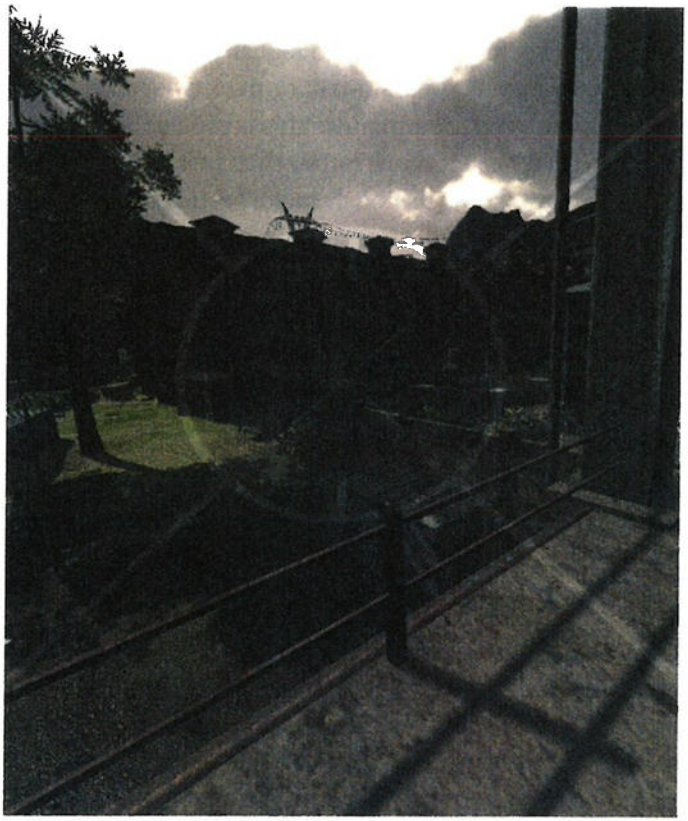


Figure 26.4 **A heads-up display in the head reference frame. No matter where the user looks with the head, the cues are always visible.** (Image courtesy of NextGen Interactions)

26.4 Speech and Gestures

The usability of speech and gestures depends upon the number and complexity of the commands. More commands require more learning—the number of voice commands and gestures should be limited to keep interaction simple and learnable.

Voice interfaces and gesture recognition systems are normally invisible to the user. Use explicit signifiers, such as a list of possible commands or icons of gestures, in the users' view so they know and remember what is possible. Neither speech nor gesture recognition is perfect. In many cases it is appropriate to have users verify commands to confirm the system understands correctly before taking action. Feedback should also be provided to let the user know a command has been understood (e.g., highlight the signifier if the corresponding command has been activated).

Meta Platforms, Inc.
Exhibit 1067
Page 317

Use a set of well-defined, natural, easy-to-understand, and easy-to-recognize gestures/words. Pushing a button to signal to the computer that a word or gesture is intended to start (i.e., push-to-talk or push-to-gesture) can keep the system from recognizing unintended commands. This is especially true when the user is also communicating with other humans, rather than just the system itself (for both voice and gestures as humans subconsciously make gestures as they talk).

26.4.1 Gestures

A **gesture** is a movement of the body or body part whereas a **posture** is a single static configuration. Each conveys some meaning whether intentional or not. Postures can be considered a subset of gestures (i.e., a gesture over a very short period of time or a gesture with imperceptible movement). Dynamic gestures consist of one or more tracked points (consider making a gesture with a controller) whereas poses require multiple tracked points (e.g., a hand pose).

Gestures can communicate four types of information (Hummels and Stappers, 1998).

Spatial information is the spatial relationship that a gesture refers to. Such gestures can manipulate (e.g., push/pull), indicate (e.g., point or draw a path), describe form (e.g., convey size), describe functionality (e.g., twisting motion to describe twisting a screw), or use objects. Such direct interaction is a form of structural communication (Section 1.2.1) and can be quite effective for VR interaction due to its direct and immediate effect on objects.

Symbolic information is the sign that a gesture refers to. Such gestures can be concepts like forming a V shape with the fingers, waving to say hello or goodbye, and explicit rudeness with a finger. The formation of such gestures is structural communication (Section 1.2.1) whereas the interpretation of the gesture is indirect communication (Section 1.2.2). Symbolic information can be useful for both human-computer interaction and human-human interaction.

Pathic information is the process (thinking, doing) the gesture is used with (e.g., subconsciously talking with one's hands). Pathic information is most commonly visceral communication (Section 1.2.1) added on to indirect communication (Section 1.2.1) that is useful for human-human interaction.

Affective information is the emotion a gesture refers to. Such gestures are more typically body gestures that convey mood such as distressed, relaxed, or enthusiastic. Affective information is a form of visceral communication (Section 1.2.1) most often used for human-human interaction, although pathic information is less commonly recognized with computer vision as discussed in Section 35.3.3.

Meta Platforms, Inc.
Exhibit 1067
Page 318

In the real world, hand gestures often augment communication with gestures such as okay, stop, size, silence, kill, goodbye, pointing, etc. Many early VR systems used gloves as input and gestures to indicate similar commands. Advantages of gestures include flexibility, the number of degrees of freedom of the human hand, the lack of having to hold a device in the hand, and not necessarily having to see (or at least look directly at) the hand. Gestures, like voice, can also be challenging due to having to remember them and most current systems have low recognition rates for more than a few gestures. Although gloves are not as comfortable, they are more consistent than camera-based systems due to not having line-of-sight issues. Push-to-gesture systems can drastically reduce false positives. This is especially true when the user is communicating with other humans, rather than just the system itself.

Direct vs. Indirect Gestures

Direct gestures are immediate and structural (Section 1.2.1) in nature and convey spatial information; they can be interpreted and responded to by the system as soon as the gesture starts. Direct manipulation, such as pushing an object, and selection via hand pointing are examples of direct gestures. **Indirect gestures** indicate more complex semantic meaning over a period of time so the start of the gesture is not sufficient—the application interprets over a range of movement so there is a delay from the start of the gesture. Indirect gestures convey symbolic, pathic, and affective information. A single pose command is somewhere between direct and indirect because the system response is immediate but not structural (the pose is interpreted as a command).

26.4.2 Speech Recognition

Speech recognition translates spoken words into textual and semantic form. If implemented well, voice commands have many advantages including keeping the head and hands free to interact while giving commands to the system. Voice recognition does have significant challenges including limited recognition capability, not always obvious command options, difficulty in selecting from a continuous scale, background noise, variability between speakers, and distraction to other individuals (McMahan et al., 2014). Regardless, speech can work well for multimodal interactions (Section 26.6). Speech recognition categories, strategies, and errors are discussed below as described by Hannema (2001).

Speech Recognition Categories

Speech recognition is often categorized into the following groups.

Speaker-independent speech recognition has the flexibility to recognize a small number of words from a wide range of users. This type of speech recognition is used with telephone navigation systems and is best used with VR when there are

Meta Platforms, Inc.
Exhibit 1067
Page 319

only a small number of options provided to the user (a VR system should visually show available commands so the user knows what the options are).

Speaker-dependent speech recognition recognizes a large number of words from a single user where the system has been extensively trained to recognize words from that specific user. This type of speech recognition can work well with VR when the user has a personal system that she uses often.

Adaptive recognition is a mix of speaker-independent and speaker-dependent speech recognition. The system does not need explicit training but learns the characteristics of the specific user as he speaks. This often requires that the user corrects the system when words are misinterpreted. Use adaptive recognition when users have their own system but don't want to bother with explicitly training the voice recognizer.

Speech Recognition Strategies

Each of the speech recognition categories listed above can use one or more of the following strategies to recognize words.

Discrete/isolated strategies recognize one word at a time from a predefined vocabulary. This strategy works well when only one word is used or there is a silence between consecutive words. Examples include commands such as "save," "undo," "restart," or "freeze."

Continuous/connected strategies recognize consecutive words from a predefined vocabulary. This is more challenging to implement than a discrete/isolated strategy.

Phonetic strategies recognize individual phonemes (small, perceptually distinct sounds; Section 8.2.3), diphones (combinations of two adjacent phonemes), or triphones (combinations of three adjacent phonemes). Triphones are computationally expensive, and the system may be slow to respond due to the number of combinations that must be recognized so are rarely used.

Spontaneous/conversational strategies attempt to determine the context of the words in sentences in a way similar to what humans do. This results in a natural spoken dialogue with the computer. This strategy can be difficult to implement well.

Speech Recognition Errors

There are several reasons why speech recognition is difficult. By being aware of the common types of errors listed below, the system can be better designed to minimize

Meta Platforms, Inc.
Exhibit 1067
Page 320

such errors. Errors can also be reduced by using a microphone designed for speech recognition (Section 27.3.3).

Deletion/rejection occurs when the system fails to match or recognize a word from the predetermined vocabulary. The advantage of this type of error is that the system recognizes the error and can request the user to repeat the word.

Substitution occurs when the system misrecognizes a word for a different word than that which was intended. If the error is not caught, the system might execute a wrong command. This error is difficult to detect, but statistical measures can be used to calculate confidence.

Insertion occurs when an unintended word is recognized. This most often occurs when the user is not intentionally speaking to the system, such as when thinking out loud or when speaking to another human. Similar to a substitution error, this can execute an unintended command. Requiring the user to push a button (e.g., a push-to-talk interface) can drastically reduce this type of error.

Context

The specific context that the user is engaged in at any particular time can help improve accuracy and better match the user's intention. This is important as words can be homonyms (the same word with multiple meanings, e.g., volume can be a volume of space or audio volume) or homophones (different words with the same sound, e.g., die and dye). Context-sensitive systems with a large vocabulary can be implemented by having the system only able to recognize a subset of that vocabulary at any particular time.

26.5 Modes and Flow

Although ideally the same metaphors should be applied across all interactions for a single application, this is often not possible. Complex applications with different types of tasks may require different interaction techniques. In such a case, different techniques might be combined together. The mechanism to choose a technique may be as simple as holding a different button or a mode selection from a hand-held panel, or the technique may be order dependent (e.g., a specific manipulation technique only occurs after a specific selection technique). Whatever the mode, that mode should be made clear to the user.

All interactions should also integrate and flow together well. The overall usability of a system depends on the seamless integration of various tasks and techniques provided by the application. One way to think about flow is to consider the sequence

Meta Platforms, Inc.
Exhibit 1067
Page 321

of basic actions. People may more often verbally state commands with the action coming before the object to be acted upon, but they tend to think about the object first. Objects are more concrete in nature so they are easier to first think about, whereas verbs are more abstract and are more easily thought about when being applied to something. For example, someone might think "pick up the book," but before thinking about picking up the book the person must first perceive and think about the book. Users prefer object-action sequences over action-object sequences as it requires less mental effort (McMahan and Bowman, 2007; McMahan, 2007). Thus, when designing interaction techniques, the selection of the object to be acted upon should be performed (at least in most cases) before taking action upon that object. The interaction technique should also enable easy and smooth transition between selecting an object and manipulating or using that object.

At a higher level, the flow of longer interactions should occur without distractions so the user can give full attention to the primary task. Ideally, users should not have to physically (whether the eyes, head, or hands) or cognitively move between tasks. Lightweight mode switching, physical props, and multimodal techniques can help to maintain the flow of interaction.

26.6 Multimodal Interaction

No single sensory input or output is appropriate for all situations. **Multimodal interactions** combine multiple input and output sensory modalities to provide the user with a richer set of interactions. The "put-that-there" interface is known as the first human-computer interface to effectively and naturally mix voice and gesture (Bolt, 1980). Note although "put-that-there" is an action-object sequence, as discussed above in Section 26.5, better flow often occurs by first selecting the object to be moved. The better implementation might be called a "that-moves-there" interface.

When choosing or designing multimodal interactions, it can be helpful to consider different ways of integrating the modalities together. Input can be categorized into six types of combinations: specialized, equivalence, redundancy, concurrency, complementarity, and transfer (Laviola, 1999; Martin, 1998). All of the input modality types are multimodal other than specialized.

Specialized input modalities limit input options to a single modality for a specific task due to the nature of the task and the design of the system. Specialization is ideal when there is clearly a single best modality for the task. For example, for some environments, selecting an object might only be performed by pointing.

Meta Platforms, Inc.
Exhibit 1067
Page 322

Equivalent input modalities provide the user a choice of which input to use, even though the result would be the same across modalities. Equivalence can be thought of as the system being adaptable to user preferences. For example, a user might be able to create the same objects either by voice or through a panel.

Redundant input modalities take advantage of two or more simultaneous actions to perform a single command. Redundancy can reduce noise and ambiguous signals, resulting in increased recognition rates. For example, a user might select a red cube with the hand while saying "select the red cube" or physically move an object with the hand while saying "move."

Concurrent input modalities enable users to issue different commands simultaneously, and thus enable users to be more efficient. For example, a user might be pointing to fly while verbally requesting information about an object in the distance.

Complementarity input modalities merge information together into a single command. Complementarity often results in faster interactions as the different modalities are typically close in time or even concurrent. For example, to delete an object, the application might require the user to move the object behind the shoulder while saying "delete." Another example is a "put-that-there" interface (Bolt, 1980) that merges voice and gesture to place an object.

Transfer occurs when information from one input modality is transferred to another input modality. Transfer can improve recognition and enable faster interactions. A user may achieve part of a task by one modality but then determine a different modality would be more appropriate to complete the task. In such a case, transfer would prevent the user from needing to start over. An example is verbally requesting a specific menu to appear, which can be interacted with by speaking or pointing. Another example is a "push-to-talk" interface. Transfer is most appropriate to use when hardware is unreliable or does not work well in some situations.

26.7 Beware of Sickness and Fatigue

Some interaction techniques, especially those controlling the viewpoint, can cause motion sickness. When choosing or creating navigation techniques, designers should carefully understand and consider scene motion and motion sickness as discussed in Part III. If motion sickness is a primary concern, then changing the viewpoint

Meta Platforms, Inc.
Exhibit 1067
Page 323

should only occur through one-to-one mapping of real head motion or teleportation (Section 28.3.4).

Some users are not comfortable looking at interfaces close to the face for extended periods of time due to the accommodation-vergence conflict (Section 13.1) that occurs in most of today's HMDs. Visual interfaces close to the face should be minimized.

As mentioned in Section 14.1, gorilla arm can be a problem for interactions that require the user to hold their hands up high and out in front of themselves for more than a few seconds at a time. This occurs even with bare-hand systems (Section 27.2.5) where the user is not carrying any additional weight. Interactions should be designed to minimize holding the hands above the waist for more than a few seconds at a time. For example, shooting a ray from the hand held at the hip is quite comfortable.

26.8 Visual-Physical Conflict and Sensory Substitution

Most VR experiences offer little touch feedback, and when they do the feedback is quite limited compared to the sense of touch in the real world. Not having full haptic feedback is more of a problem than just not feeling objects. The hand or other body part (or physical device) continues to move through the object since there is no (or limited) physical force stopping it from doing so. As a result, the physical location of the hand may no longer match the visual location.

Enforcing simulated physics so the hand does not visually pass through visual geometry is often preferred by users when the penetrations are only slight (shallow penetration). When deeper penetration occurs, then users prefer the visual hands to match the physical hands even though that breaks the intuition that hands do not pass through objects (Lindeman, 1999). Stopping the visual hand for deep penetration can be especially confusing when the visual hand pops out of a different part of the penetrated object than where the visual hand has previously been visually stopped. A compromise solution for non-realistic interactions is to draw two hands when the physical hand and physically simulated hand diverge (see ghosting below).

In some cases, the virtual hand can be considerably offset from the physical hand without the user noticing as visual representation tends to dominate proprioception (Burns et al., 2006). However, this is not always the case. Vision is generally stronger than proprioception when moving the hand in a left/right and/or up/down direction, but proprioception can be stronger when moving the hand in depth (forward/back) (Van Beers et al., 2002).

Sensory substitution is the replacement of an ideal sensory cue that is not available with one or more other sensory cues. Examples of sensory substitution that work well with VR are described below.

Meta Platforms, Inc.
Exhibit 1067
Page 324

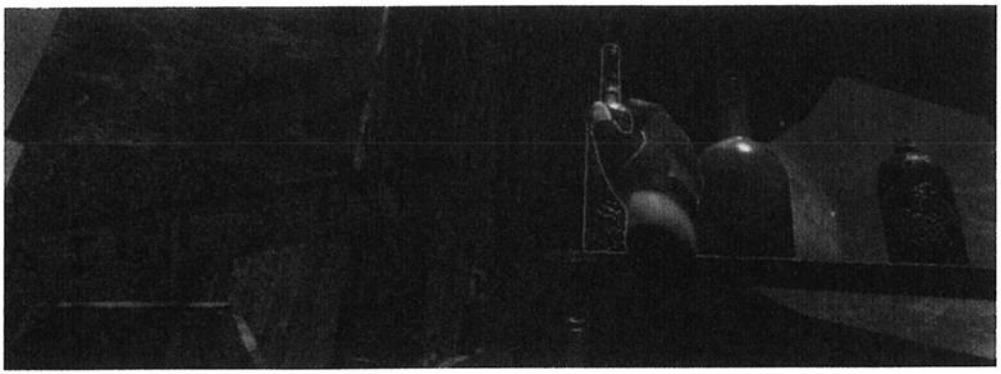


Figure 26.5 **In the game "The Gallery: Six Elements," the bottle is highlighted to show the object can be grabbed.** (Image courtesy of Cloudhead Games)

Ghosting is a second simultaneous rendering of an object in a different pose than the actual object. In some cases it is appropriate to render the hand twice—both where the physical hand is located and where the physics simulation states the hand is located. Ghosting is also often used to provide a clue of where a virtual object will be snapped into place if released. Be careful of using ghosting for training applications as users can depend on ghosting as a crutch that will not be available for the real-world task.

Highlighting is visually outlining or changing the color of an object. Highlighting is most often used to show the hand has intersected with an object so that it can be selected or picked up. Highlighting is also used to convey that an object is able to be selected or grabbed when the hand is close even though a collision has not yet occurred. Figure 26.5 shows an example of highlighting.

Audio cues are very effective in conveying to a user that one of his hands has collided with some geometry. Audio might be as simple as a tone sound or real-world recorded audio track. In some cases, providing multiple audio files with variations (e.g., random grunt sounds when colliding with a virtual wall or when shot by an enemy) can help with a sense of realism and reduce annoyance. Continuous contact sounds can also be used to convey sliding along surfaces. Sound properties as pitch or amplitude might also change depending on penetration depth.

Passive haptics (static physical objects that can be touched; Section 3.2.3) are effective when the virtual world is limited to the physical space where no virtual

Meta Platforms, Inc.
Exhibit 1067
Page 325

navigation can occur (i.e., when the real-world reference frames and virtual-world reference frames are consistent; Section 26.3) or when tracked physical tools travel with the user (i.e., the physical and virtual objects are spatially compliant; Section 25.2.5). Because vision often dominates proprioception, perfect spatial compliance is not always required (Burns et al., 2006). Redirected touching warps virtual space to map many differently shaped virtual objects onto a single real object (i.e., hand or finger tracking is not one-to-one) in a way that the discrepancy between virtual and physical is below the user's perceptual threshold (Kohli, 2013). For example, when one's real hand traces a physical object, the virtual hand can trace a slightly differently shaped virtual object.

Rumble causes a hand-held controller to vibrate. Although not the same haptic force that would occur in the real world, rumble feedback can be quite an effective cue for informing the user she has collided with an object.

Meta Platforms, Inc.
Exhibit 1067
Page 326

Input Devices

Input devices are the physical tools/hardware used to convey information to the application and to interact with the virtual environment. Some interaction techniques work more or less across different input devices whereas other techniques work only with input devices that have specific characteristics. Thus, appropriately choosing input hardware that best fits the application's interaction techniques is an important design decision (or, conversely, designing and implementing interaction techniques depends upon available input hardware).

This chapter describes some general characteristics of input devices and then describes the primary classes of input devices.

27.1 Input Device Characteristics

Input devices can be very different, and the characteristics of each should be considered when choosing hardware and designing interactions.

27.1.1 Size and Shape

The most obvious characteristics to a new VR user are the basic shape and size of the input device. The shape and size has more to do with just how the controller looks and feels in the hand. Large hand-held devices are primarily controlled by large muscle groups of the shoulder, elbow, and wrist, whereas smaller hand-held devices utilize smaller and faster muscle groups in the fingers (Bowman et al., 2004). Smaller devices can also decrease clutching—the releasing and regrasping of an object in order to complete a task due to not being able to complete it in a single motion (such as a wrench). Gloves also use these smaller muscle groups and have the advantage of being able to freely touch and feel other items.

27.1.2 Degrees of Freedom

Input devices are often classified by the number of degrees of freedom (DoF) they report. **Degrees of freedom** are the number of parameters that an input device is

Meta Platforms, Inc.
Exhibit 1067
Page 327

capable of manipulating (also see Section 25.2.3). Devices range from a single DoF (e.g., an analog trigger), to 6 DoF that measure full 3D translation (up/down, left/right, forward/backward) and rotation (roll, pitch, and yaw), to full hand or body tracking with many DoFs. A traditional mouse, joystick, trackball, and touchpad (a rotatable ball that is essentially an upside-down mouse) are examples of 2 DoF devices. VR hand tracking should have a minimum of 6 DoF (multiple points tracked on the hand have more than 6 DoF). For a majority of active VR experiences, one or more 6 DoF hand-held controllers is often the most appropriate choice. For some simple tasks only requiring navigation and no direct interaction, a non-tracked hand-held controller is good enough.

27.1.3 Relative vs. Absolute

Relative input devices measure the differences between the current and last measurement. Mice, trackballs, and inertial trackers are examples of relative devices. Relative devices drift over time and thus are not nulling compliant (Section 25.2.5). Although limited, the Nintendo Wii proved relative devices can work well for natural interactions under some circumstances if the applications are carefully designed. VR relative devices typically use inertial measurement units (IMUs) that have the advantage of having a higher update rate (e.g., 1,000 Hz) and faster response (e.g., 1 ms) than absolute measurements. **Absolute input devices** measure relative to a constant point of reference independent of past measurements and are nulling compliant. **Hybrid tracking systems** fuse both relative and absolute trackers to provide the advantages of both. VR head and hand tracking should be absolute (although relative devices can arguably estimate absolute pose of the hands from modeling the constraints imposed by the physicality of the arms and hand).

27.1.4 Separable vs. Integral

Integral input devices enable users to control all DoFs simultaneously from a single motion (a single composition) whereas **separable input devices** contain at least one DoF that cannot be controlled simultaneously from a single motion (two or more distinct compositions). A gamepad with two different analog sticks is an example of a separable device. A 2D device that enables control of more than two dimensions via mode switching is also an example of a separable device. VR hand tracking should be integral.

27.1.5 Isometric vs. Isotonic

Isometric input devices measure pressure or force with no actual movement. **Isotonic input devices** measure deflection from a center point and may or may not have some

Meta Platforms, Inc.
Exhibit 1067
Page 328

resistance. Mice are isotonic input devices. Joysticks can be either isotonic or isometric. Isotonic input devices are best for controlling position, whereas isometric input devices are best for controlling rates such as navigation speed. An isometric joystick, for example, works well for controlling velocity (i.e., hold to continue moving).

27.1.6 Buttons

Buttons control one DoF via pushing with a finger and typically take on one of two states (e.g., pushed or not pushed) although some buttons can take on analog values (also known as analog triggers). Buttons are often used to change modes, to select and object, or to start an action. Although buttons can be useful for VR applications, too many buttons can lead to confusion and error—especially when the button-mapping functionality is unclear or inconsistent (although attaching labels to virtual controllers can help—see Figure 26.3 as an example). Consider the capabilities and intuitiveness of desktop applications that are controlled via no more than three buttons on a mouse.

There is a debate between bare-hand system (Section 27.2.5) and hand-held controller (Sections 27.2.2 and 27.2.3) advocates as to the utility of buttons. For example, Microsoft Kinect and Leap Motion developers believe buttons are a primitive and unnatural form of input whereas Playstation Move and Sixense Stem developers believe buttons are an essential part of game play. Like any great debate, the answer is "it depends" (Jerald et al., 2012). Buttons can be abstracted to indirectly trigger nearly any action, but this abstraction can cause a disconnect between the user and the application. Buttons are most effective when an action is binary, when the action needs to occur often, when reliability is required, and when physical feedback to the user is essential. Buttons are also ideal for time-sensitive actions, since very little time is required for the physical action (i.e., an entire dynamic gesture does not need to be completed in order to register the action). Gestures can be slower and more fatiguing than button presses, particularly in command-intensive tasks such as modeling or radiology. Natural buttonless hand manipulation is most effective for providing a sense of realism and presence, when abstraction is not appropriate, or when detailed tracking of the entire hand is required.

27.1.7 Encumbrance

Unencumbered input devices do not require physical hardware to be held or worn. Such systems are implemented with camera systems. Thus no "suit-up" time is required (although calibration might be necessary). Unencumbered systems can also reduce hygiene issues (Section 14.4) since a physical device is not passed between users. Non-encumbrance is not always a design goal; holding something physically in

Meta Platforms, Inc.
Exhibit 1067
Page 329

the hand can add to the sense of presence (Section 3.2.3); consider holding a controller in the hand vs. not holding a controller in the hand for a shooting or golf experience.

27.1.8 Ability to Fully Interact with Physical Objects

Some devices enable one to touch the real world in natural ways without the device getting in the way. Bare-hand tracking systems (i.e., camera systems) and gloves are the most common examples of this. Held devices and world-grounded devices (Section 27.2.1) must be released before the hand can fully interact with other physical objects. In such cases, the physical devices should be tracked and rendered in the virtual world so the user can reach out to grab them again.

27.1.9 Device Reliability

Device reliability is the extent to which an input device can consistently work within the user's entire personal space (and a larger volume if the user is expected to physically move around). Devices should ideally have 100% reliability anywhere the user can reach with no loss in tracking acquisition. Reliability should carefully be considered when choosing an input device as unreliable devices can result in frustration, fatigue (e.g., due to having to hold the hand high and in front of the body; Section 14.1), increased cognitive load (e.g., because the user must think about holding the device in a certain way), breaks-in-presence (Section 4.2), and reduced performance.

Unreliable tracking can be due to multiple reasons and divided into two groups: (1) implementation limitations and (2) inherent physical limitations. An inherent physical limitation is the best a device will ever be able to achieve given an optimal engineering solution/implementation. Some devices cannot possibly provide 100% reliability no matter what the engineering effort. A system that requires line of sight from a sensor to the tracked device can be occluded by another physical object (such as a hand or torso), in which case there is no way for the system to reliably determine the pose of the device (although the state of the device can be estimated for short periods of time). VR devices should ideally work in all orientations and hand postures (e.g., hands covering sensors or tracking the fingers when a fist is made).

Another challenge of reliability occurs for users attempting to work in a tracked volume smaller than the user's personal space. An example is a vision-based system with a limited field of view. Lighting can also be a challenge for some vision-based systems, especially in uncontrolled environments outside of the laboratory or some other highly controlled space. Other systems only recognize gestures when the hand is oriented or held in a certain way. Many camera-based hand tracking systems only reliably recognize poses and gestures when the hands are perpendicular to the camera with the fingers visible.

Meta Platforms, Inc.
Exhibit 1067
Page 330

27.1.10 Haptics Capable

Active haptics can easily be added to physical devices that are worn or held. However, the degree of haptics might be limited depending on the size of the device and if attached to the world in some way (Section 3.2.3).

27.2 Classes of Hand Input Devices

The most important VR input devices are the human hands, and this section explores how different types of devices can integrate the hands into VR. An **input device class** is a set of input devices that share the same essential characteristics that are crucial to interaction. Input device classes described in this section focus on the hands and are classified as world-grounded input devices, non-tracked hand-held controllers, tracked hand-held controllers, hand-worn devices, and bare-hand controllers.

Table 27.1 summarizes some of the most essential characteristics of the hand input device classes described in this section and non-hand input device classes described in Section 27.3. As can be seen, no single input device class is universally advantageous although some classes can be combined to create a hybrid system with more advantages. For example, a bare-hands camera-based system might be used with tracked hand-held controllers (although simultaneous usage can be difficult due to different characteristics, so bare hands might be used until the controller is picked up). Note the table is based upon inherent physical limitations rather than today's existing implementations. Technical specifications such as update rate, latency, etc. are not included here as such specs are independent of the hardware (e.g., a fast update rate could potentially be implemented on any device class).

27.2.1 World-Grounded Input Devices

World-grounded input devices, like world-grounded haptics (Section 3.2.3), are designed to be constrained or fixed in the real world and are most often used to interact with desktop systems.

Keyboards and mice are considered to be world-grounded devices and are the most popular form of input that works extremely well for its intended task—2D desktop manipulation. However, such input is not a good way of interacting for most all immersive applications (with the possible exception of video-see-through augmented reality where users can see the mouse and keyboard).

Trackballs and joysticks mounted to a permanent location are world-grounded devices. Other devices offer up to 6 DoF through pushing, pulling, twisting, and/or buttons for mode changes. However, these devices suffer from similar limitations for VR as the mouse due to not being designed to be held comfortably and freely

Meta Platforms, Inc.
Exhibit 1067
Page 331

Table 27.1 **Comparison of hand and non-hand input device classes.**

	Proprioception	Consistent	Usable in Lap or the Side	Haptics Capable	Unencumbered	Physical Buttons	Hands Free to Interact with Real World	General Purpose
Hand Input Device Class								
World-Grounded Devices	✓	✓		✓	✓	✓	✓	
Non-Tracked Hand-Held Controllers		✓	✓	✓		✓		
Bare Hands	✓				✓		✓	✓
Tracked Hand-Held Controllers	✓	✓	✓	✓		✓		✓
Hand Worn	✓	✓	✓	✓		✓	✓	✓
Non-Hand Input Device Class								
Head Tracking	✓	✓					✓	✓
Eye Tracking							✓	
Microphone			✓		✓		✓	✓
Full-Body Tracking	✓	✓	✓	✓			✓	✓
Treadmills	✓	✓			✓		✓	

in the hands. There are exceptions (e.g., to mount a joystick on the arm of a chair or if simulating a desktop environment where the physical controls precisely match a virtual desktop), but from a human-centered design perspective there should be a solid reason to choose such devices, instead of "because it is available" or "it is what someone else is using."

World-grounded devices that work extremely well for VR are specialty devices such as handlebars, steering wheels, gas and brake pedals, cockpits, and automotive interior controls. Such devices can be especially good for travel, as most users already

Meta Platforms, Inc.
Exhibit 1067
Page 332

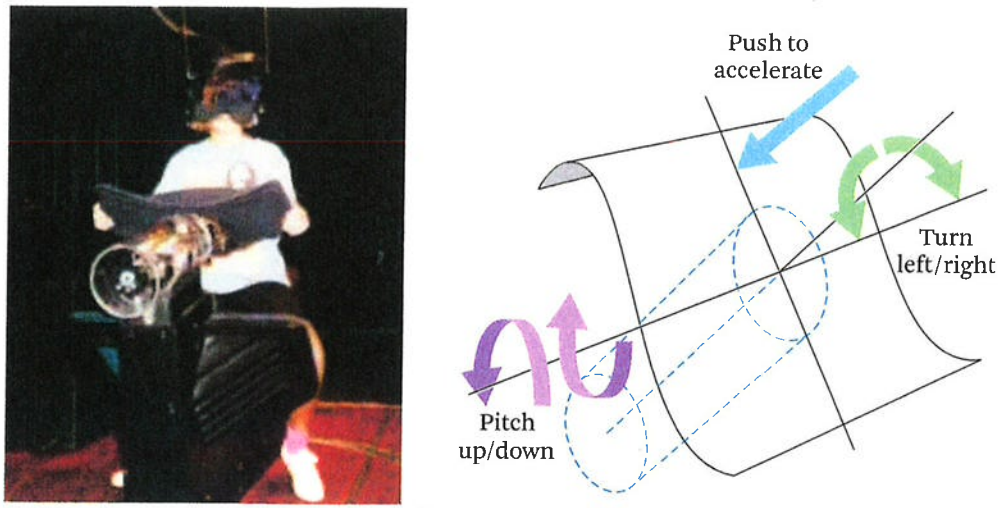


Figure 27.1 **The Disney Aladdin world-grounded input device along with its mapping for viewpoint control.** (From Pausch et al. (1996))

have real-world experience with devices. Even if world-grounded devices aren't actually used in the real world, they can still be quite effective and presence inducing if designed well. For example, Disney's Aladdin Magic Carpet Ride, 3 DoF controls (Figure 27.1) provides an intuitive physical interface for travel. One reason such controls are so effective is due to having physical signifiers, affordances, and feedback, e.g., the user feels what can be done and how she is doing it. Some challenges of these devices are that creators can't assume there is a wide user base that owns such hardware and it can be difficult to generalize the devices to work with a wide range of tasks. Thus, such devices are more commonly used at location-based entertainment venues where large groups of people use the same device(s) and the device can be designed or modified for the particular VR experience.

27.2.2 Non-Tracked Hand-Held Controllers

Non-tracked hand-held controllers are devices held in the hand that include buttons, joysticks/analog sticks, triggers, etc. but are not tracked in 3D space. Traditional video game input devices such as joysticks and gamepads are the most common form of non-tracked hand-held controllers (Figure 27.2). Many VR applications are starting to support such game controllers. These controllers work much better than the mouse and keyboard since the controller can be held comfortably in the lap where users can continually hold the controller. Many gamers have an intuitive feel of where the

Meta Platforms, Inc.
Exhibit 1067
Page 333



Figure 27.2 **The Xbox One controller is an example of a non-tracked hand-held controller.** (Image courtesy of Stock photo C juniorbeep)

buttons are through years of use. Controllers with analog sticks work surprisingly well for navigating within VR (Section 28.3.2).

Although not tracked, such controllers can increase presence for seated experiences by placing visual hands and controllers at the approximate location of the user's lap since most users hold the controller in such a position. The visual controller and hands have also been found to cause users to subconsciously move the hands and controller to the visual controller (Andrew Robinson and Sigurdur Gunnarsson, personal communication, May 11, 2015). Unfortunately, when a user moves his hands away from the assumed position, a break-in-presence typically occurs if the user sees the virtual hands stay in place.

27.2.3 Tracked Hand-Held Controllers

Tracked hand-held controllers are typically 6 DoF devices (known as "wands" in the VR research community where they have been used for decades) and can also contain functionality offered by non-tracked hand-held controllers. Tracked hand-held controllers are currently the best option for a majority of interactive VR applications. Tracked hand-held controllers are easy to use for many 3D tasks due to their natural, direct mapping to hand motion. Because the controllers are tracked, they can be

Meta Platforms, Inc.
Exhibit 1067
Page 334



Figure 27.3 **The Sixense STEM (left) and Oculus Touch (right) tracked hand-held controllers.** (Image courtesy of Sixense)

both visually co-located with the real hands (i.e., spatially and temporally compliant; Section 25.2.5), as well as physically felt, providing proprioceptive and passive haptics/touch cues. Labels can also be attached to the virtual representation to provide immediate instruction of what the buttons do by simply looking at where the hands are physically located (Section 26.3.4 and Figure 26.3), adding a big advantage over traditional desktop and gamepad input. Viewpoint manipulation with these devices is typically achieved using buttons and a trackball, integrated analog sticks (as on the Sixense STEM and Oculus Touch controllers as shown in Figure 27.3), or by flying with the hands. Such techniques are described in Section 28.3. Other types of physical controls and feedback can also be added to these devices, such as trackpads and active haptics (e.g., vibration).

Tracked hand-held controllers have the advantage of acting as a physical prop, which enhances presence through physical touch. Not only do such controls facilitate communication with the virtual world, but they also help to make spatial relationships seem more concrete to the user (Hinckley et al., 1998). However, such props come at the cost of not being able to directly/fully touch and feel other passive objects in the world and world-grounded input devices, such as seats, handlebars, and cockpit controls (Section 27.2), without first setting down the controller/prop.

Hand-held tracked devices typically use inertial, electromagnetic, ultrasonic, or optical (camera) technologies. Each of these technologies has advantages and disadvantages, and hand-held trackers ideally use some hybrid method of integrating multiple technologies together (sensor fusion) to provide both high precision and accuracy.

Meta Platforms, Inc.
Exhibit 1067
Page 335

27.2.4 Hand-Worn Devices

Hand-worn input devices include gloves, muscle-tension sensors (electromyographic or EMG sensors), such as what Thalmic Labs recently made popular with the Myo (which is worn on the arm but measures hand motion), and rings.

Many believe gloves (Figure 27.4) to be the ultimate VR interface as they theoretically have many advantages, such as not having line-of-sight, sensor field-of-view, or lighting requirements so the hands can be held comfortably to the side or in the lap with no concern of losing tracking, resulting in less gorilla arm if the interaction techniques are designed well (Section 18.9). Like bare hands, gloves also have the advantage that the hands and fingers can still fully interact with other physical objects.

Unfortunately, like bare-hand systems, full-hand tracked gloves are lacking in their current form and will require dramatic improvement to be used by the masses. Consistent recognition of more than a few gestures is still challenging due to the lack of consistent finger tracking accuracy. Recognizing more than a few gestures requires the user to recalibrate often due to the glove moving on the hand. Gloves must also be put on and worn, which can become uncomfortable and result in sweaty hands. There is also a risk of social resistance to wearing gloves similar to the resistance of Google Glass—although those willing to wear an HMD on their face are unlikely to care what others think of wearing gloves. If such challenges can be solved, then gloves may eventually become the input device of choice for VR.

The Fakespace Pinch Gloves have functions more like buttons, with near 100% consistent recognition, more than typical gloves that do full hand and finger tracking. They work via a conductive cloth sewn into the tip of each finger. When two or more fingers touch then the circuit closes resulting in a signal. This simple design provides the capability for a large number of pinch gestures; combinations range from two to ten fingers touching each other plus poses involving separate but simultaneous pinches (e.g., left thumb to left index finger and right thumb to right index finger pinched at the same time). In practice, due to the usability of physical hand constraints and users' willingness to memorize gestures, applications will only use so many of these gestures, similar to how too many buttons on a hand controller would be too confusing. Pinch gloves can be used quite well with many of the example techniques described in Chapter 28.

Perhaps one of the most significant advantages of gloves is that both full hand tracking and button simulation via pinch gestures can be combined such as demonstrated by LaViola and Zeleznik (1999). Haptics can also be used with gloves, such as is done with the CyberGlove CyberTouch (as shown in Figure 27.4 but with buzzers added to provide a sense of haptics).

Meta Platforms, Inc.
Exhibit 1067
Page 336



Figure 27.4 **The CyberGlove is an example of a hand-worn device.** (Image courtesy of CyberGlove Systems LLC)

If EMG sensors and rings can be made more accurate, then they too might become an ideal fit for many applications.

27.2.5 Bare Hands

Bare-hand systems work via sensors aimed at the hands (mounted in the world or on the HMD). Figure 27.5 shows the hands and a skeletal model fit to the hands, as seen by the user. The obvious major advantage is that the user's hands are completely unencumbered. Many believe bare-hand systems will ultimately be the ideal VR interface. Although the bare hands work extremely well in the real world, it turns out consistent interaction with the bare hands in VR is an enormous challenge. Challenges include not having a sense of touch, fatigue from holding the hands in front of the sensor, line-of-sight requirements, and consistent recognition of gestures across a wide range of users. Such technical challenges lead to usability challenges, such as being able to comfortably work with the hands in the lap without concern for where the sensor(s) is located. The bare hands also lack physical buttons, which is important for some applications but not important for other applications (Section 27.1.6).

Regardless of the challenges of effectively working with the bare hands in VR, seeing the entirety of one's hands in 3D is extremely compelling and provides a nice sense of presence for the periods of time when the tracking does consistently work. It remains to be seen if such challenges will be overcome and/or accepted by a wide range of users.

27.3 Classes of Non-hand Input Devices

VR input can occur through more than just the hands. This section describes head tracking, eye tracking, microphones, and full-body tracking.

Meta Platforms, Inc.
Exhibit 1067
Page 337

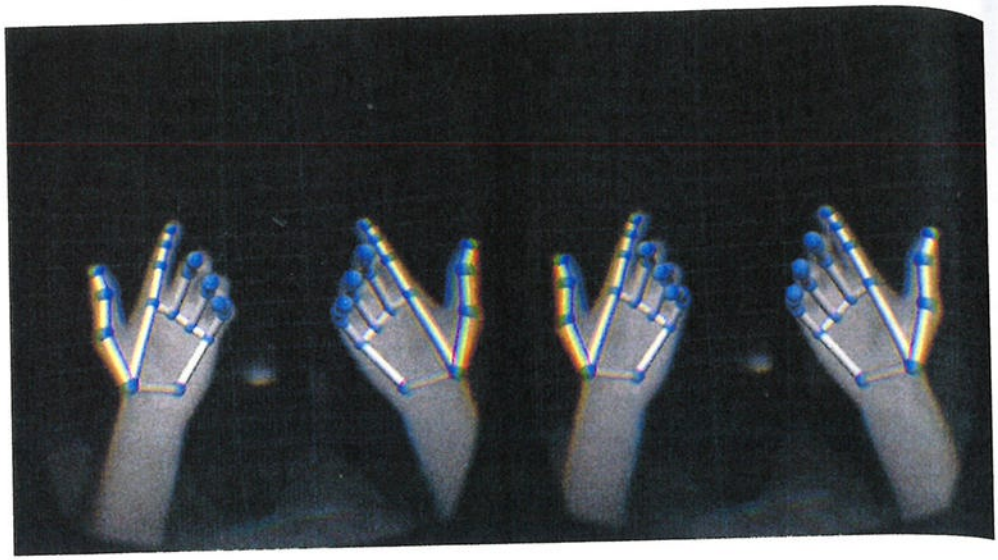


Figure 27.5 **A depth-sensing camera on the HMD looking out enables one to precisely see one's own hands. Here a skeleton model is also fit to the hands.** (Image courtesy of Leap Motion)

27.3.1 Head Tracking

Head tracking must be accurate, precise, fast, and well calibrated for the virtual world to appear stable. World stability is essential for VR and assumed to work well, but is not the focus of this chapter. Here, head-tracking interaction refers to interaction that modifies or provides feedback beyond just seeing the virtual environment. The most common form of head-tracking interaction is to aim by looking. One way of doing this is to provide a reticle or pointer in the middle of the screen (Section 26.3.5) that is triggered by a button press, firing in the direction of the reticle or selecting an option the reticle projects onto. Other more subtle interactions can be used, such as having an action occur in the direction the user is looking, having characters respond when looked at, or simple head gestures such as nodding the head yes or no.

27.3.2 Eye Tracking

Eye-tracking input for VR is a largely unexplored topic other than the obvious of selecting (Section 28.1.2) or firing where one is looking as mostly demonstrated with interactive eye-tracking systems that are integrated within some of today's HMDs.

The Midas Touch problem refers to the fact that people expect to look at things without that look "meaning" something (Jacob, 1991). Interacting via eye tracking alone is usually not a good idea—eye tracking works better with multimodal input.

Meta Platforms, Inc.
Exhibit 1067
Page 338

For example, signaling via a clutch (e.g., push a button, blink, or say "select") typically works better than signaling via a dwell time. Even with a clutch, designing interactions that work well with gaze is a challenge. Straightforward feedback in the form of a pointer/reticle that moves with the eyes can be annoying to users and occludes viewing in the part of vision with the highest visual acuity. Eye saccades can also make the pointer jitter or jump in unintended ways. These problems can be mitigated by only turning on the pointer when a button is held down and filtering out high-frequency motions.

Eye tracking can be more effective for specialized tasks and for subtle interactions, such as how a character responds when looked at. The following guidelines are useful to consider when designing eye-tracking interactions (Kumar, 2007).

Maintain the natural function of the eyes. Our eyes are meant for looking, and interaction designers should maintain the natural function of the eye. Using the eyes for other purposes overloads the visual channel.

Augment rather than replace. Attempting to replace existing interfaces with eye tracking is typically not appropriate. Instead, think about how to add functionality onto already existing and newley created interfaces. Gaze can provide context and inform the system that the user is paying attention to a specific object or area in the scene.

Focus on interaction design. Focus on the overall experience instead of eye tracking alone. Consider the number of steps in the interaction, the amount of time it takes, the cost of an error/failure, cognitive load, and fatigue.

Improve the interpretation of eye movements. Gaze data is noisy. Consider how to best filter eye movements, classify gaze data, recognize gaze patterns, and take other input modalities into account.

Choose appropriate tasks. Don't try to force gaze to solve every problem. Eye tracking is not appropriate for all tasks. Consider the task and scenario before choosing gaze.

Use passive gaze over active gaze. Consider ways in which gaze can be used more passively so the eyes can better maintain their natural function.

Leverage gaze information for other interactions. Leverage the system's knowledge of where the user is paying attention in order to provide context for non-gaze interactions.

Although typically not interactive, eye tracking will have significant benefit for VR usability by providing clues to content creators of what most engages a user's attention

Meta Platforms, Inc.
Exhibit 1067
Page 339

(see attention maps in Section 10.3.2), similar to the use of eye tracking to inform and enhance website design.

27.3.3 Microphones

A **microphone** is an acoustic sensor that transforms physical sound into an electric signal. Use a headset microphone specifically designed for speech recognition that includes noise-canceling features that remove/reduce ambient noise. The microphone should be able to be easily adjusted/positioned while the HMD is on and while not being able to see it. Accuracy improves as the microphone comes closer to and in front of the mouth. Microphones should be comfortable and light.

To prevent speech recognition errors (Section 26.4.2), such as sounds picked up when thinking aloud or another person's voice, use a push-to-talk interface. If handheld controllers are used, then the push-to-talk button should be located on the controller.

27.3.4 Full-Body Tracking

Full-body tracking consists of tracking more than just the head, hands, and torso. Full-body tracking can significantly add to the illusion of self-embodiment as well as the illusion of social presence (Section 4.3). Tracking a large number of features can also be used to enhance interaction (e.g., a game that allows kicking a ball).

VR full-body tracking is typically done with a motion capture suit, similar to what is used in the film industry. Different suits utilize different technologies such as electromagnetic sensors, retro-reflective markers, and inertial sensors. Depth cameras such as Microsoft Kinect can theoretically track the entire body, but it can be difficult to capture the entire body unless a large number of cameras are used. Regardless, full-body camera capture systems, like Microsoft Kinect, can be used for creating extremely compelling experiences, even though resolution is not yet great and much of the body might go in and out of view. Figure 27.6 shows real-time capture of the real world and display of the resulting point-cloud data within an HMD.

Meta Platforms, Inc.
Exhibit 1067
Page 340

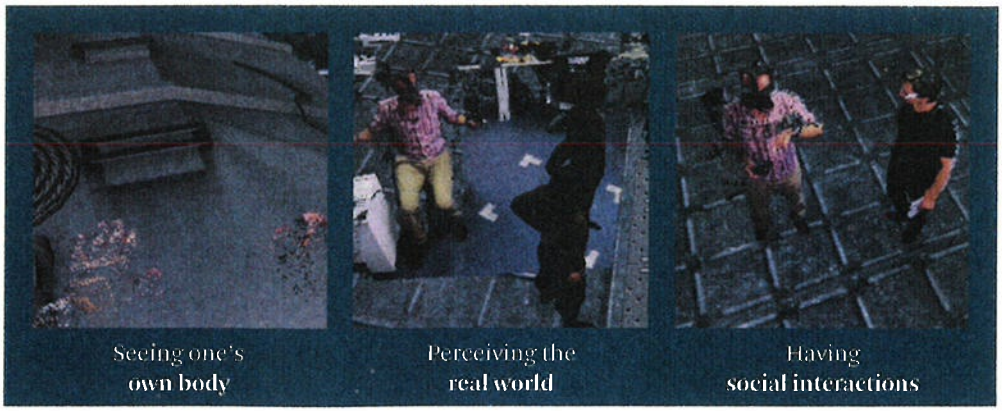


Figure 27.6 **Depth cameras enable users to see their own body, the real world, and/or other people from the real world.** (Image courtesy of Dassault Systèmes, iV Lab)

Meta Platforms, Inc.
Exhibit 1067
Page 341

Interaction Patterns and Techniques

An **interaction pattern** is a generalized high-level interaction concept that can be used over and over again across different applications to achieve common user goals. The interaction patterns here are intended to describe common approaches to general VR interaction concepts at a high level. Note interaction patterns are different than software design patterns (Section 32.2.5) that many system architects are familiar with. Interaction patterns are described from the user's point of view, are largely implementation independent, and state relationships/interactions between the user and the virtual world along with its perceived objects.

An **interaction technique** is more specific and more technology dependent than an interaction pattern. Different interaction techniques that are similar are grouped under the same interaction pattern. For example, the walking pattern (Section 28.3.1) covers several walking interaction techniques ranging from real walking to walking in place. The best interaction techniques consist of high-quality affordances, signifiers, feedback, and mappings (Section 25.2) that result in an immediate and useful mental model for users.

Distinguishing between interaction patterns and interaction techniques is important for multiple reasons (Sedig and Parsons, 2013).

- There are too many existing interaction techniques with many names and characterizations to remember, and many more will be developed in the future.
- Organizing interaction techniques under the umbrella of a broader interaction pattern makes it easier to consider appropriate design possibilities by focusing on conceptual utility and higher-level design decisions before worrying about more specific details.
- Broader pattern names and concepts make it easier to communicate interaction concepts.
- Higher-level groupings enable easier systematic analysis and comparison.

Meta Platforms, Inc.
Exhibit 1067
Page 342

- When a specific technique fails, then other techniques within the same pattern can be more easily thought about and explored, resulting in better understanding of why that specific interaction technique did not work as intended.

Both interaction patterns and interaction techniques provide conceptual models to experiment with, suggestions and warnings of use, and starting points for innovative new designs. Interaction designers should know and understand these patterns and techniques well so they have a library of options to choose from depending on their needs and a base to innovate upon. Do not fall into the trap that there is a single best interaction pattern or technique. Each pattern and technique has strengths and weaknesses depending on application goals and the type of user (Wingrave et al., 2005). Understanding distinctions and managing trade-offs of different techniques is essential to creating high-quality interactive experiences.

This chapter divides VR interaction patterns into selection, manipulation, viewpoint control, indirect control, and compound patterns. The interaction patterns and their interaction techniques as organized in this chapter are shown in Table 28.1. The

Table 28.1 Interaction patterns as organized in this chapter. More specific example interaction techniques are described for each interaction pattern.

- Selection Patterns
  - Hand Selection Pattern
  - Pointing Pattern
  - Image-Plane Selection Pattern
  - Volume-Based Selection Pattern
- Manipulation Patterns
  - Direct Hand Manipulation Pattern
  - Proxy Pattern
  - 3D Tool Pattern
- Viewpoint Control Patterns
  - Walking Pattern
  - Steering Pattern
  - 3D Multi-Touch Pattern
  - Automated Pattern
- Indirect Control Patterns
  - Widgets and Panels Pattern
  - Non-Spatial Control Pattern
- Compound Patterns
  - Pointing Hand Pattern
  - World-in-Miniature Pattern
  - Multimodal Pattern

Meta Platforms, Inc.
Exhibit 1067
Page 343

first four patterns are often used sequentially (e.g., a user may travel toward a table, select a tool on the table, and then use that tool to manipulate other objects on the table) or can be integrated together into compound patterns. Interaction techniques that researchers and practitioners have found to be useful are then described within the context of the broader pattern. These techniques are only examples, as many other techniques exist, and such a list could never be exhaustive. The intent for describing these patterns and techniques is for readers to directly use them, to extend them, and to serve as inspiration for creating entirely new ways of interacting within VR.

28.1 Selection Patterns

Selection is the specification of one or more objects from a set in order to specify an object to which a command will be applied, to denote the beginning of a manipulation task, or to specify a target to navigate toward (McMahan et al., 2014). Selection of objects is not necessarily obvious in VR, especially when most objects are located at a distance from the user. **Selection patterns** include the Hand Selection Pattern, Pointing Pattern, Image-Plane Selection Pattern, and Volume-Based Selection Pattern. Each has advantages over the others depending on the application and task.

28.1.1 Hand Selection Pattern

Related Patterns

Direct Hand Manipulation Pattern (Section 28.2.1) and 3D Tool Pattern (Section 28.2.3).

Description

The **Hand Selection Pattern** is a direct object-touching pattern that mimics real-world interaction—the user directly reaches out the hand to touch some object and then triggers a grab (e.g., pushing a button on a controller, making a fist, or uttering a voice command).

When to Use

Hand selection is ideal for realistic interactions.

Limitations

Physiology limits a fully realistic implementation of hand selection (the arm can only be stretched so far and the wrist rotated so far) to those objects within reach (personal space), requiring the user to first travel to place himself close to the object to be selected. Different user heights and arm lengths can make it uncomfortable for some people to select objects that are at the edge of personal space. Virtual hands and

Meta Platforms, Inc.
Exhibit 1067
Page 344



Figure 28.1 **A realistic hand with arm (left), semi-realistic hands with no arms (center), and abstract hands (right).** (Image courtesy of NextGen Interactions)

arms often occlude objects of interest and can be too large to select small items. Non-realistic hand selection techniques are not as limiting.

Exemplar Interaction Techniques

Realistic hands. **Realistic hands** is extremely compelling for providing an illusion of self-embodiment (Section 4.3). Although ideally the entire arm would be tracked, inverse kinematics can estimate the pose of the arm quite well where users typically don't notice differences in arm pose if the head and/or torso is tracked along with the hand. Figure 28.1 (left) shows an example of a view from a user who has grabbed a bottle.

Modeling users (e.g., measuring arm length) and placing objects within a comfortable range depending on the measured arm length is ideal. However, Digital ArtForms found no complaints from ages 10 to adult in a semi-immersive world after setting a single arm-length scale value that was reasonable for the entire range of body sizes.

Non-realistic hands. Hands do not need to necessarily look real, and trying to make hands and arms realistic can limit interaction. **Non-realistic hands** does not try to mimic reality but instead focuses on ease of interaction. Often hands are used without arms (Figure 28.1, center) so that reach can be scaled to make the design of interactions easier. Although the lack of arms can be disturbing at first, users quickly learn to accept having no arms. The hands also need not look like hands. For abstract applications, users are quite accepting of abstract 3D cursors (Figure 28.1, right) and still feel like they are directly selecting objects. Such hand cursors reduce problems of visual occlusion. Hand occlusion can also be mitigated by making the hand transparent (although proper transparency rendering of multiple objects can be technically challenging to do correctly).

Meta Platforms, Inc.
Exhibit 1067
Page 345

Go-go technique. The **go-go technique** (Poupyrev et al., 1996) expands upon the concept of a non-realistic hand by enabling one to reach far beyond personal space. The virtual hand is directly mapped to the physical hand when within 2/3 of the full arm's reach and when extended further, the hand "grows" in a nonlinear manner enabling the user to reach further into the environment. This technique enables closer objects to be selected (and manipulated) with greater accuracy while allowing further objects to be easily reached. Physical aspects of arm length and height have been found to be important for the go-go technique, so measuring arm length should be considered when using this technique (Bowman, 1999). Measuring arm length can be done by simply asking the user to hold out the hands in front of the body at the start of the application. Bowman and Hodges (1997) describes extensions to the go-go technique, such as providing rate control (i.e., velocity) options that enable infinite reach, and compares these to pointing techniques. Non-isomorphic hand rotations (Section 28.2.1) are similar but scale rotations instead of position for manipulation.

28.1.2 Pointing Pattern

Related Patterns

Widgets and Panels Pattern (Section 28.4.1) and Pointing Hand Pattern (Section 28.5.1).

Description

The Pointing Pattern is one of the most fundamental and often-used patterns for selection. The **Pointing Pattern** extends a ray into the distance and the first object intersected can then be selected via a user-controlled trigger. Pointing is most typically done with the head (e.g., a crosshair in the center of the field of view) or a hand/finger.

When to Use

The Pointing Pattern is typically better than the Hand Selection Pattern unless realistic interaction is required. This is especially true for selection beyond personal space and when small hand motions are desired. Pointing is faster when speed of remote selection is important (Bowman, 1999), but is also often used for precisely selecting close objects, such as pointing with the dominant hand to select components on a panel (Section 28.4.1) held in the no-ndominant hand.

Limitations

Selection by pointing is usually not appropriate when realistic interaction is required (the exception might be if a laser pointer or remote control is being modeled).

Meta Platforms, Inc.
Exhibit 1067
Page 346

Straightforward implementations result in difficulty selecting small objects in the distance. Pointing with the hand can be imprecise due to natural hand tremor (Bowman, 1999). Object snapping and precision mode pointing described below can mitigate this problem.

Exemplar Interaction Techniques

Hand pointing. When hand tracking is available, **hand pointing** with a ray extending from the hand or finger is the most common method of selection. The user then provides a signal to actually select the item of interest (e.g., a button press or gesture with the other hand).

Head pointing. When no hand tracking is available, selection via head pointing is the most common form of selection. **Head pointing** is typically implemented by drawing a small pointer or reticle at the center of the field of view so the user simply lines up the pointer with the object of interest and then provides a signal to select the object. The signal is most commonly a button press but when buttons are not available then the item is often selected by dwelling, that is, by holding the pointer on the object for some defined period of time. Selection via dwelling is not ideal due to having to wait for objects to be selected and accidental selection when looking at an object of interest.

Eye gaze selection. Eye gaze **selection** is a form of pointing implemented with eye tracking (Section 27.3.2). The user simply looks at an item of interest and then provides a signal to select the looked-at object. In general, eye gaze selection is typically not a good selection technique, primarily due to the Midas Touch problem discussed in Section 27.3.2.

Object snapping. **Object snapping** (Haan et al., 2005) works by objects having scoring functions that cause the selection ray to snap/bend toward the object with the highest score. This technique works well when selectable objects are small and/or moving.

Precision mode pointing. **Precision mode pointing** (Kopper et al., 2010) is a non-isomorphic rotation technique that scales down the rotational mapping of a pointer, called the C/D (control/display) ratio. The result is a "slow motion cursor" that enables fine pointer control. A zoom lens can also be used that scales the area around the cursor to enable seeing smaller objects (but the zoom should not be affected by head pose unless the zoom area on the display is small; Section 23.2.5). The user can control the amount of zoom with a scroll wheel on a hand-held device.

Meta Platforms, Inc.
Exhibit 1067
Page 347

Two-handed pointing. Two-handed pointing originates the selection at the near hand and extends the ray through the far hand (Miné et al., 1997). This provides more precision when the hands are further apart and fast rotations about a full 360° range when the hands are closer together (whereas a full range of 360° for a single hand pointer is difficult due to physical hand constraints). The distance between the hands can also be used to control the length of the pointer.

28.1.3 Image-Plane Selection Pattern

Also Known As

Occlusion and Framing.

Related Pattern

World-in-Miniature Pattern (Section 28.5.2).

Description

The **Image-Plane Selection Pattern** uses a combination of eye position and hand position for selection (Pierce et al., 1997). This pattern can be thought about as the scene and hand being projected onto a 2D image plane in front of the user (or on the eye). The user simply holds one or two hands between the eye and the object of interest and then provides a signal to select the object when the object lines up with the hand and eye.

When to Use

Image-plane techniques simulate direct touch at a distance, thus are easy to use (Bowman et al., 2004). These techniques work well at any distance as long as the object can be seen.

Limitations

Image-plane selection works for a single eye, so users should close one eye while using these techniques (or use a monoscopic display). Image-plane selection results in fatigue when used often due to having to hold the hand up high in front of the eye. As in the Hand Selection Pattern, the hand often occludes objects if not transparent.

Exemplar Interaction Techniques

Head crusher technique. In the **head crusher technique** (Figure 28.2), the user positions his thumb and forefinger around the desired object in the 2D image plane.

Sticky finger technique. The **sticky finger technique** provides an easier gesture—the object underneath the user's finger in the 2D image is selected.

Meta Platforms, Inc.
Exhibit 1067
Page 348

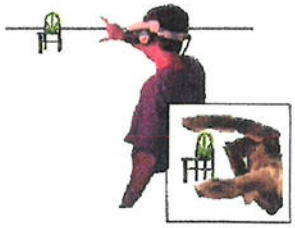


Figure 28.2 **The head crusher selection technique. The inset shows the user's view of selecting the chair.** (From Pierce et al. (1997))

Lifting palm technique. In the **lifting palm technique**, the user selects objects by flattening his outstretched hand and positions the palm so that it appears to lie below the desired object.

Framing hands technique. The **framing hands technique** is a two-handed technique where the hands are positioned to form the two corners of a frame in the 2D image surrounding an object.

28.1.4 Volume-Based Selection Pattern

Related Patterns

3D Multi-Touch Pattern (Section 28.3.3) and World-in-Miniature Pattern (Section 28.5.2).

Description

The **Volume-Based Selection Pattern** enables selection of a volume of space (e.g., a box, sphere, or cone) and is independent of the type of data being selected. Data to be selected can be volumetric (voxels), point clouds, geometric surfaces, or even space containing no data (e.g., to follow with filling that space with some new data). Figure 28.3 shows how a selection box can be used to carve out a volume of space from a medical dataset.

When to Use

The Volume-Based Selection Pattern is appropriate when the user needs to select a not-yet-defined set of data in 3D space or to carve out space within an existing dataset. This pattern enables selection of data when there are no geometric surfaces (e.g., medical CT datasets), whereas geometric surfaces are required for implementing many other selection patterns/techniques (e.g., pointing requires intersecting a ray with object surfaces).

Meta Platforms, Inc.
Exhibit 1067
Page 349

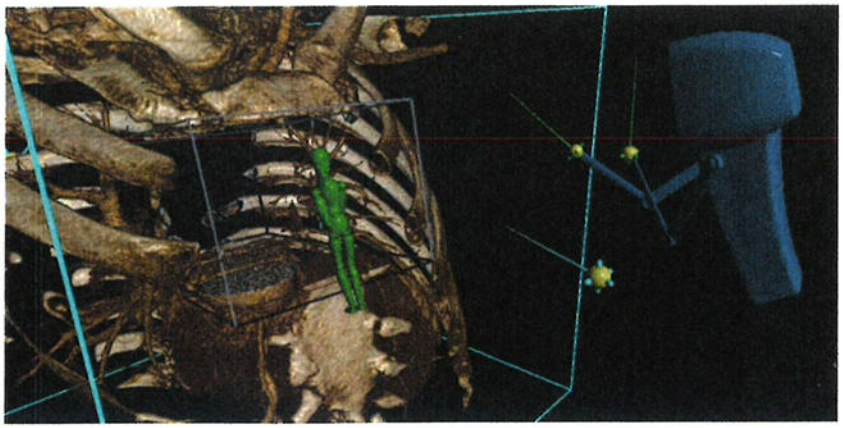


Figure 28.3 **The blue user/avatar on the right has carved out a volume of space within a medical dataset via snapping, nudging, and reshaping a selection box (the gray box in front of the green avatar). The green avatar near the center is stepping inside the dataset to examine it from within.** (Image courtesy of Digital ArtForms)

Limitations

Selecting volumetric space can be more challenging than selecting a single object with the other more common selection patterns.

Exemplar Interaction Techniques

Cone-casting flashlight. The **cone-casting flashlight technique** uses pointing, but instead of using a ray, a cone is used. This results in easier selection of small objects than standard pointing via ray casting. If the intent is a single object, then the object closest to the cone's center line or the object closest to the user can be selected (Liang and Green, 1994). A modification of this technique is the aperture technique (Forsberg et al., 1996), which enables the user to control the spread of the selection volume by bringing the hand closer or further away.

Two-handed box selection. **Two-handed box selection** uses both hands to position, orient, and shape a box via snapping and nudging. Snap and nudge are asymmetric techniques where one hand controls the position and orientation of the selection box, and the second hand controls the shape of the box (Yoganandan et al., 2014). Both snap and nudge mechanisms have two stages of interaction—grab and reshape. Grab influences the position and orientation of the box. Reshape changes the shape of the box.

Meta Platforms, Inc.
Exhibit 1067
Page 350

Snap immediately brings the selection box to the hand and is designed to quickly access regions of interest that are within arm's reach of the user. Snap is an absolute interaction technique, i.e., every time snap is initiated the box position/orientation is reassigned. Therefore, snap focuses on setting the initial pose of the box and placing it comfortably within arm's reach.

Nudge enables incremental and precise adjustment and control of the selection box. Nudge works whether the selection box is near or far away from the user by maintaining the box's current position, orientation, and scale for the initial grab, but subsequent motion of the box is locked to the hand. Once attached to the hand, the box is positioned and oriented relative to its initial state—because of this, nudge can be thought of as a relative change in box pose. The box can then be simultaneously reshaped with the other hand while holding down a button.

28.2 Manipulation Patterns

Manipulation is the modification of attributes for one or more objects such as position, orientation, scale, shape, color, and texture. Manipulation typically follows selection, such as the need to first pick up an object before throwing it. **Manipulation Patterns** include the Direct Hand Manipulation Pattern, Proxy Pattern, and 3D Tools Pattern.

28.2.1 Direct Hand Manipulation Pattern

Related Patterns

Hand Selection Pattern (Section 28.1.1), Pointing Hand Pattern (Section 28.5.1), and 3D Tool Pattern (Section 28.2.3).

Description

The **Direct Hand Manipulation Pattern** corresponds to the way we manipulate objects with our hands in the real world. After selecting the object, the object is attached to the hand moving along with it until released.

When to Use

Direct positioning and orientation with the hand have been shown to be more efficient and result in greater user satisfaction than other manipulation patterns (Bowman and Hodges, 1997).

Limitations

Like the Hand Selection Pattern (Section 28.1.1), a straightforward implementation is limited by the physical reach of the user.

Meta Platforms, Inc.
Exhibit 1067
Page 351

Exemplar Interaction Techniques

Non-isomorphic rotations. Some form of clutching is required to rotate beyond certain angles, and clutching can hinder performance due to wasted motion (Zhai et al., 1996). Clutching can be reduced by using non-isomorphic rotations (Poupyrev et al., 2000) that allow one to control larger ranges of 3D rotation with smaller wrist rotation. Non-isomorphic rotations can also be used to provide more precision by mapping large physical rotations to smaller virtual rotations.

Go-go technique. The go-go technique (Section 28.1.1) can be used for manipulation as well as selection with no mode change.

28.2.2 Proxy Pattern

Related Patterns

Direct Hand Manipulation Pattern (Section 28.2.1) and World-in-Miniature Pattern (Section 28.5.2).

Description

A **proxy** is a local object (physical or virtual) that represents and maps directly to a remote object. The **Proxy Pattern** uses a proxy to manipulate a remote object. As the user directly manipulates the local object(s), the remote object(s) is manipulated in the same way.

When to Use

This pattern works well when a remote object needs to be intuitively manipulated as if it were in the user's hands or when viewing and manipulating objects at multiple scales (e.g., the proxy object can stay the same size relative to the user even as the user scales himself relative to the world and remote object).

Limitations

The proxy can be difficult to manipulate as intended when there is a lack of directional compliance (Section 25.2.5); that is, when there is an orientation offset between the proxy and the remote object.

Exemplar Interaction Technique

Tracked physical props. **Tracked physical props** are objects directly manipulated by the user (a form of passive haptics; Section 3.2.3) that map to one or more virtual objects and are often used to specify spatial relationships between virtual objects. Hinckley et al. (1998) describe an asymmetric two-handed 3D neurosurgical visualization system where the non-dominant hand holds a doll's head and the dominant hand

Meta Platforms, Inc.
Exhibit 1067
Page 352



Figure 28.4 **A physical proxy prop used to control the orientation of a neurological dataset.** (From Hinckley et al. (1994))

holds a planar object or a pointing device (Figure 28.4). The doll's head directly maps to a remotely viewed neurological dataset and the planar object controls a slicing plane to see inside the dataset. The pointing device controls a virtual probe. Such physical proxies provide direct action-task correspondence, facilitate natural two-handed interactions, provide tactile feedback to the user, and are extremely easy to use without requiring any training.

28.2.3 3D Tool Pattern

Related Patterns

Hand Selection Pattern (Section 28.1.1) and Direct Hand Manipulation Pattern (Section 28.2.1).

Description

The **3D Tool Pattern** enables users to directly manipulate an intermediary 3D tool with their hands that in turn directly manipulates some object in the world. An example of a 3D tool is a stick to extend one's reach or a handle on an object that enables the object to be reshaped.

When to Use

Use the 3D Tool Pattern to enhance the capability of the hands to manipulate objects. For example, a screwdriver provides precise control of an object by mapping large rotations to small translations along a single axis.

Meta Platforms, Inc.
Exhibit 1067
Page 353

Limitations

3D tools can take more effort to use if the user must first travel and maneuver to an appropriate angle in order to apply the tool to an object.

Exemplar Interaction Techniques

Hand-held tools. **Hand-held tools** are virtual objects with geometry and behavior that are attached to the user's hand. Such tools can be used to control objects from afar (like a TV remote control) or to work more directly on an object. A paintbrush used to draw on a surface of an object is an example of a hand-held tool. Hand-held tools are often easier to use and understand than widgets (Section 28.4.1) due to being more direct.

Object-attached tools. An **object-attached tool** is a manipulable tool that is attached/colocated with an object. Such a tool results in a more coupled signifier representing the affordance between the object, tool, and user. For example, a color icon might be located on an object, and the user simply selects the icon at which point a color cube appears so the user can choose the color of the object. Or if the shape of a box can be changed, then an adjustment tool can be made available on the corners of the box (e.g., dragging the vertex).

Jigs. Precisely aligning and modeling objects can be difficult with 6 DoF input devices due to having no physical constraints. One way to enable precision is to add virtual constraints with jigs. **Jigs**, similar to real-world physical guides used by carpenters and machinists, are grids, rulers, and other referenceable shapes that the user attaches to object vertices, edges, and faces. The user can adjust the jig parameters (e.g., grid spacing) and snap other objects into exact position and orientation relative to the object the jig is attached to. Figure 28.5 shows some examples of jigs. **Jig kits** support the snapping together of multiple jigs (e.g., snapping a ruler to a grid) for more complex alignments.

28.3 Viewpoint Control Patterns

Viewpoint control is the task of manipulating one's perspective and can include translation, orientation, and scale. Travel (Section 10.4.3) is a form of viewpoint control that does not allow scaling.

Controlling one's viewpoint is equivalent to moving, rotating, or scaling the world. For example, moving the viewpoint to the left is equivalent to moving the world to the right, or scaling oneself to be smaller is equivalent to scaling the world to be larger.

Meta Platforms, Inc.
Exhibit 1067
Page 354

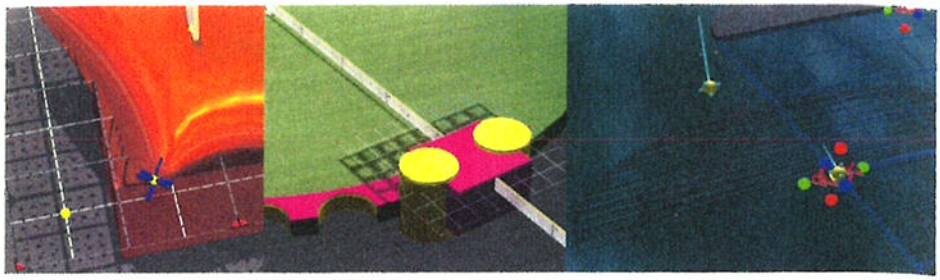


Figure 28.5 **Jigs used for precision modeling. The blue 3D crosshairs in the left image represent the user's hand. The user drags the lower left corner of the orange object to a grid point (left). The user cuts shapes out of a cylinder at 15° angles (center). The user precisely snaps a wireframe-viewed object onto a grid (right).** (Image courtesy of Digital ArtForms and Sixense)

Thus, users perceive themselves either as moving through the world (self-motion) or as the world moving around them (world motion) as the viewpoint changes.

Viewpoint Control Patterns include the Walking Pattern, Steering Pattern, 3D Multi-Touch Pattern, and Automated Pattern. Warning: Some implementations of these patterns may induce motion sickness and are not appropriate for users new to VR or those who are sensitive to scene motion. In many cases, sickness can be reduced by integrating the techniques with the suggestions in Part III.

28.3.1 Walking Pattern

Description

The **Walking Pattern** leverages motion of the feet to control the viewpoint. Walking within VR (Steinicke et al., 2013) includes everything from real walking to mimicking walking by moving the feet when seated.

When to Use

This pattern matches or mimics real-world locomotion and therefore provides a high degree of interaction fidelity. Walking enhances presence and ease of navigation (Usoh et al., 1999) as well as spatial orientation and movement understanding (Chance et al., 1998). Real walking is ideal for navigating small to medium-size spaces, and such travel results in no motion sickness if implemented well.

Limitations

Walking is not appropriate when rapid or distant navigation is important. True walking across large distances requires a large tracked space, and for wired headsets cable

Meta Platforms, Inc.
Exhibit 1067
Page 355

tangling can pull on the headset and be a tripping hazard (Section 14.3). A human spotter should closely watch walking users to help stabilize them if necessary. Fatigue can result with prolonged use, and walking distance is limited by the physical motion that users are willing to endure. Cable tangling is also an issue when using a wired system; often an assistant holds the wires and follows the user to prevent tangling and tripping.

Exemplar Interaction Techniques

Real walking. **Real walking** matches physical walking with motion in the virtual environment; the mapping from real to virtual is one-to-one. Because of high interaction fidelity, real walking is an ideal interface for many VR experiences. Real walking typically does not directly measure foot motion, but instead tracks the head. Real walking results in less motion sickness due to better matching visual and vestibular cues (although sickness can still result due to issues such as latency and system miscalibration). The motion of one's feet in a self-embodied avatar can be estimated or the feet can be tracked to provide greater biomechanical symmetry. Unfortunately, real walking by itself limits travel in the virtual world to the physically tracked space.

Redirected walking. **Redirected walking** (Razzaque et al., 2001) is a technique that allows users to walk in a VR space larger than the physically tracked space. This is accomplished by rotational and translational gains that are different than the true motion of the user, directing the user away from the edges of tracked space (or physical obstacles). Ideally, the gains are below perceptible thresholds so that the user does not consciously realize he is being redirected.

Walking in place. Various forms of walking in place exist (Wendt, 2010), but all consist of making physical walking motions (e.g., lifting the legs) while staying in the same physical spot but moving virtually. Walking in place works well when there is only a small tracked area and when safety is a primary concern. The safest form of walking in place is for the user to be seated. Theoretically users can walk in place for any distance. However, travel distances are limited by the physical effort the users are willing to make. Thus, walking in place works well for small and medium-size environments where only short durations of travel are required.

The human joystick. The **human joystick** (McMahan et al., 2012) utilizes the user's position relative to a central zone to create a 2D vector that defines the horizontal direction and velocity of virtual travel. The user simply steps forward to control speed. The human joystick has the advantage that only a small amount of tracked space is required (albeit more than walking in place).

Meta Platforms, Inc.
Exhibit 1067
Page 356

Treadmill walking and running. Various types of treadmills exist for simulating the physical act of walking and running (Section 3.2.5). Although not as realistic as real walking, treadmills can be quite effective for controlling the viewpoint, providing a sense of self-motion, and walking an unlimited distance. Techniques should make sure foot direction movement is compliant with forward visual motion. Otherwise, treadmill techniques that lack directional and temporal compliance (Section 25.2.5) can be worse than no treadmill. Treadmills with safety harnesses are ideal, especially when physical running is required.

28.3.2 Steering Pattern

Description

The **Steering Pattern** is continuous control of viewpoint direction that does not involve movement of the feet. There is typically no need to control viewpoint pitch with controllers as is done with desktop systems since users can physically look up and down.

When to Use

Steering is appropriate for traveling great distances without the need for physical exertion. When exploring, viewpoint control techniques should allow continuous control, or at least the ability to interrupt a movement after it has begun. Such techniques should also require minimum cognitive load so the user can focus on spatial knowledge acquisition and information gathering. Steering works best when travel is constrained to some height above a surface, acceleration/deceleration can be minimized (Section 18.5), and real-world stabilized cues can be provided (Sections 12.3.4 and 18.2).

Limitations

Steering provides less biomechanical symmetry than the walking pattern. Many users report symptoms of motion sickness with steering. Virtual turning is more disorienting than physical turning.

Exemplar Interaction Techniques

Navigation by leaning. **Navigation by leaning** moves the user in the direction of the lean. The amount of lean typically maps to velocity. One advantage of this technique is no requirement for hand tracking. Motion sickness can be significant as velocity varies (i.e., acceleration).

Gaze-directed steering. **Gaze-directed steering** moves the user in the direction she is looking. Typically, the starting and stopping motion in the gaze direction is controlled

Meta Platforms, Inc.
Exhibit 1067
Page 357

by the user via a hand-held button or joystick. This is easy to understand and can work well for novices or for those accustomed to first-person video games where the forward direction is identical to the look direction. However, it can also be disorienting as any small head motion changes the direction of travel, and frustrating since users cannot look in one direction while traveling in a different direction.

Torso-directed steering. Torso-directed steering (also called chair-directed steering) separates out the direction of travel from the way one is looking. This has more interaction fidelity than gaze-directed steering since in the real world one does not always walk in the direction the head is pointed. In the case when the torso or chair is not tracked, a general forward direction can be assumed. This technique can have more of a nauseating effect if the user does not have a mental model of what the forward direction is or when the entire body is turned when the torso or chair is not tracked. Visual cues can be provided to help users maintain a sense of the forward direction (Figure 18.2).

One-handed flying. One-handed flying works by moving the user in the direction the finger or hand is pointing. Velocity can be determined by the distance of the hand from the head.

Two-handed flying. Two-handed flying works by moving the user in the direction determined by the vector between the two hands, and the speed is proportional to the distance between the hands (Miné et al., 1997). A minimum hand separation is considered to be a "dead zone" where motion is stopped. This enables one to quickly stop motion by quickly bringing the hands together. Flying backward with two hands is more easily accomplished than one-handed flying (which requires an awkward hand or device rotation) by swapping the location of the hands. The hands can also more comfortably control velocity at a lower position closer to the body than one-handed flying, resulting in less gorilla arm.

Dual analog sticks. Dual analog sticks (also known as joysticks or analog pads) work surprisingly well for steering over a terrain (i.e., forward/back and left/right). In most cases, standard first-person game controls should be used where the left stick controls 2D translation (pushing up/down translates the body and viewpoint forward/backward, pushing left/right translates the body and viewpoint left/right) and the right stick controls left/right orientation (pushing left rotates the body and viewing direction to the left and pushing right rotates the body and viewing direction to the right). This mapping is surprisingly intuitive and is consistent with traditional first-person video games (i.e., gamers already understand how to use such controls so they have little learning curve).

Meta Platforms, Inc.
Exhibit 1067
Page 358

Virtual rotations can be disorienting and sickness inducing for some people. Because of this, the designer might design the experience to have the content consistently in the forward direction so that no virtual rotation is required. Alternatively, if the system is wireless and the torso or chair is tracked, then there is no need for virtual rotations since the user can physically rotate 360°.

World-grounded steering devices. World-grounded input devices (Section 27.2.1) such as flight sticks or steering wheels are often used to steer through a world. Such devices can be quite effective for viewpoint control due to the sense of actively controlling a physical device.

Virtual steering devices. Instead of using physical steering devices, virtual steering devices can be used. **Virtual steering devices** are visual representations of real-world steering devices (although they do not actually physically exist in the experience) that are used to navigate through the environment. For example, a virtual steering wheel can be used to control a virtual vehicle the user sits in. Virtual devices are more flexible than physical devices as they can be easily changed in software. Unfortunately, virtual devices are difficult to control due to having no proprioceptive force feedback (although some haptic feedback can be provided when using a hand-held controller with haptic capability).

28.3.3 3D Multi-Touch Pattern

Related Patterns

World-in-Miniature Pattern (Section 28.5.2) and Volume-Based Selection Pattern (Section 28.1.4).

Description

The **3D Multi-Touch Pattern** enables simultaneous modification of the position, orientation, and scale of the world with the use of two tracked points. Similar to 2D multi-touch on a touch screen, translation via 3D multi-touch is obtained by grabbing and moving space with one hand (monomanual interaction) or with both hands (synchronous bimanual interaction). One difference from 2D multi-touch is that one of the most common ways of using 3D multi-touch is to "walk" with the hands by alternating the grabbing of space with each hand (like pulling on a rope but with hands typically wider apart). Scaling of the world is accomplished by grabbing space with both hands and moving the hands apart or bringing them closer together. Rotation of the world is accomplished by grabbing space with both hands and rotating about a point (typically either about one hand or the midpoint between the hands). Translation, rotation, and scale can all be performed simultaneously with a single two-handed gesture.

Meta Platforms, Inc.
Exhibit 1067
Page 359

When to Use

3D multi-touch works well for non-realistic interactions when creating assets, manipulating abstract data, viewing scientific datasets, or rapidly exploring large and small areas of interest from arbitrary viewpoints.

Limitations

3D multi-touch is not appropriate when the user is confined to the ground. 3D multi-touch can be challenging to implement as small nuances can affect the usability of the system. If not implemented well, 3D multi-touch can be frustrating to use. Even if implemented well, there can be a learning curve on the order of minutes for some users. Constraints, such as those that keep the user upright, limit scale, and/or disable rotations, can be added for novice users. When scaling is enabled and the display is monoscopic (or there are few depth cues), it can be difficult to distinguish between a small nearby object and a larger object that is further away. Therefore, visual cues that help the user create and maintain a mental model of the world and one's place in it can be helpful.

Exemplar Interaction Techniques

Digital ArtForms' Two-Handed Interface. Digital ArtForms has built a mature 3D multi-touch interface called THI (Two-Handed Interface) (Schultheis et al., 2012) based off of the work of Mapes and Moshell (1995) and Multigen-Paradigm's SmartScene interface (Homan, 1996) from the 1990s. Figure 28.6 shows a schematic for manipulating the viewpoint. Scale and rotation occur about the middle point between the two hands. Rotation of the world is accomplished by grabbing space with both hands and rotating

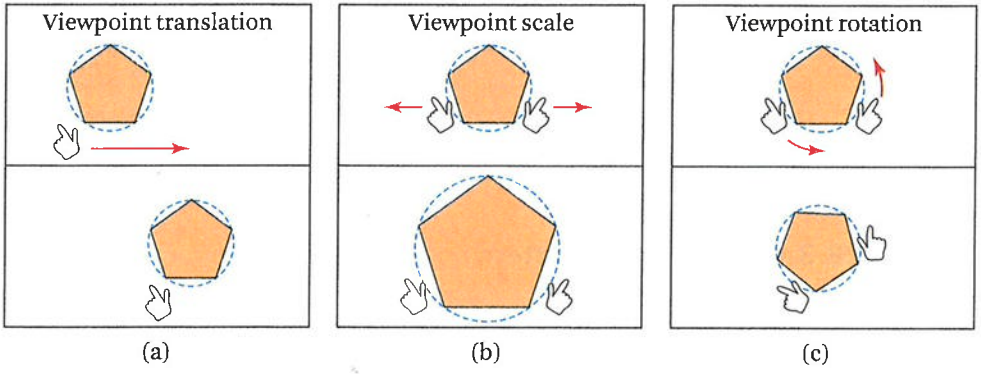


Figure 28.6 **Translation, scale, and rotation using Digital ArtForms' Two-Handed Interface.** (From Mlyniec et al. (2011))

Meta Platforms, Inc.
Exhibit 1067
Page 360

as if rotating oneself about a large tiltable valve centered at the midpoint between the hands. Note this is not the same as attaching the world to a single hand as rotating with a single hand can be quite nauseating. Translation, rotation, and scale can all be performed simultaneously with a single two-handed gesture. In multi-user environments, other users' avatars appear to grow or shrink as they scale themselves.

Once learned, this implementation of treating the world as an object works well when navigation and selection/manipulation tasks are frequent and interspersed, since viewpoint control and object control are similar other than pushing a different button (i.e., only a single interaction metaphor needs to be learned and cognitive load is reduced by not having to switch between techniques). This gives users the ability to place the world and objects of interest into personal space at the most comfortable working pose via position, rotation, and scaling operations. Digital ArtForms calls this "posture and approach" (similar to what Bowman et al. (2004) call maneuvering). Posture and approach reduces gorilla arm (Section 14.1) and users have worked for hours without reports of fatigue (Jerald et al., 2013). In addition, physical hand motions are non-repetitive by nature and are reported not to be subject to repetitive stress due to the lack of a physical planar constraint as is the case with a mouse.

The spindle. Building a mental model of the point being scaled and rotated about along with visualizing intent can be difficult for new users, especially when depth cues are absent. A "spindle" (Figure 28.7) consisting of geometry connecting the two hands (what Balakrishnan and Hinckley (2000) call visual integration), along with a visual indication of the center of rotation/scale dramatically helps users plan their actions and speeds the training process. Users simply place the center point of the spindle at the point they want to scale and rotate about, push a button in each hand, and "pull/scale" themselves toward it (or equivalently "pull/scale" the point and the world toward the user) while also optionally rotating about that point. In addition to visualizing the center of rotation/scale, the connecting geometry provides depth-occlusion cues that provide information of where the hands are relative to the geometry.

28.3.4 Automated Pattern

Description

The **Automated Pattern** passively changes the user's viewpoint. Common methods of achieving this are by being seated on a moving vehicle controlled by the computer or by teleportation.

Meta Platforms, Inc.
Exhibit 1067
Page 361

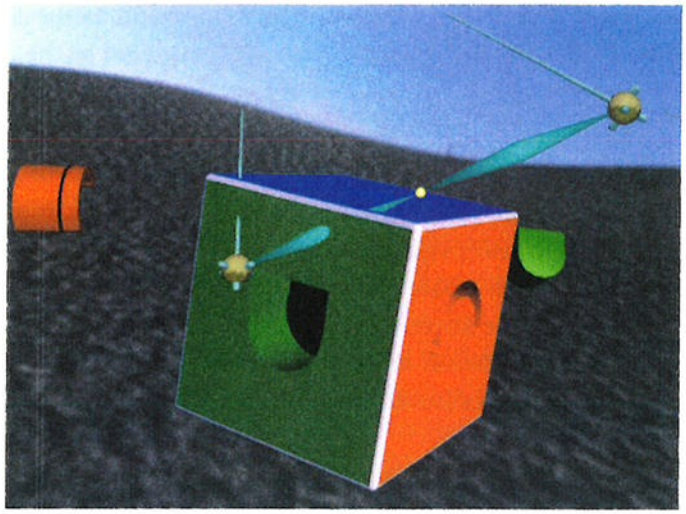


Figure 28.7 **Two hand cursors and a spindle connecting those cursors. The yellow dot between the two cursors is the point that is rotated and scaled about.** (From Schultheis et al. (2012))

When to Use

Use when the user is playing the role of a passive observer as a passenger controlled by some other entity or when free exploration of the environment is not important or not possible (e.g., due to limitations of today's cameras designed for immersive film).

Limitations

The passive nature of this technique can be disorienting and sometimes nauseating (depending on implementation). This pattern is not meant to be used to completely control the camera independent of user motion. VR applications should generally allow the look direction to be independent from the travel direction so the user can freely look around even if other viewpoint motion is occurring. Otherwise significant motion sickness will result.

Exemplar Interaction Techniques

Reducing motion sickness. Motion sickness can be significantly reduced with this technique by keeping travel speed and direction constant (i.e., keep velocity constant; Section 18.5), providing world-stabilized cues (Section 18.2), and creating a leading indicator (Section 18.4) so users know what motions to expect.

Meta Platforms, Inc.
Exhibit 1067
Page 362

Passive vehicles. Passive vehicles are virtual objects users can enter or step onto that transport the user along some path not controlled by the user. Passenger trains, cars, airplanes, elevators, escalators, and moving sidewalks are examples of passive vehicles.

Target-based travel and route planning. Target-based travel (Bowman et al., 1998) gives a user the ability to select a goal or location he wishes to travel to before being passively moved to that location. **Route planning** (Bowman et al., 1999) is the active specification of a path between the current location and the goal before being passively moved. Route planning can consist of drawing a path on a map or placing markers that the system uses to create a smooth path.

Teleportation. Teleportation is relocation to a new viewpoint without any motion. Teleportation is most appropriate when traveling large distances, between worlds, and/or when reducing motion sickness is a primary concern. Fading out and then fading in a scene is less startling than an instantaneous change. Unfortunately, straightforward teleportation comes at the cost of decreasing spatial orientation—users find it difficult to get their bearings when transported to a new location (Bowman and Hodges, 1997).

28.4 Indirect Control Patterns

Indirect Control Patterns provide control through an intermediary to modify the environment or the system. Indirect control is more abstract than selection, manipulation, and viewpoint control. Indirect control is ideal when an obvious spatial mapping does not exist or it is difficult to directly manipulate an aspect of the environment. Example uses include controlling the overall system, issuing commands, changing modes, and modifying non-spatial parameters.

Whereas the previously described techniques primarily describe both what should be done and how it is done, indirect control typically specifies only what should be done and the system determines how to do it. Because indirect control is not directly linked to that which it is controlling, signifiers such as the shape and size of controls, their visual representation and labeling, and apparent affordances of their underlying control structure are extremely important.

Indirect Control Patterns include the Widgets and Panels Pattern and Non-Spatial Control Pattern.

Meta Platforms, Inc.
Exhibit 1067
Page 363

28.4.1 Widgets and Panels Pattern

Related Patterns

Hand Selection Pattern (Section 28.1.1), Pointing Pattern (Section 28.1.2), and Direct Hand Manipulation Pattern.

Description

The **Widgets and Panels Pattern** is the most common form of VR indirect control, and typically follows 2D desktop widget and panel/window metaphors. A **widget** is a geometric user interface element. A widget might only provide information to the user or might be directly interacted with by the user. The simplest widget is a single label that only provides information. Such a label can also act as a signifier for another widget (e.g., a label on a button). Many system controls are a 1D task and thus can be implemented with widgets such as pull-down menus, radio buttons, sliders, dials, and linear/rotary menus. **Panels** are container structures that multiple widgets and tools can be placed upon. Placement of panels is important for being able to easily access them. For example, panels can be placed on an object, floating in the world, inside a vehicle, on the display, on the hand or input device, or somewhere near the body (e.g., a semicircular menu always surrounding the waist).

When to Use

Widgets and panels are useful for complex tasks where it is difficult to directly interact with an object. Widgets are normally activated via a Pointing Pattern (Section 28.1.2) but can also be combined with other selection options (e.g., select a property for objects within some defined volume). Widgets can provide more accuracy than directly manipulating objects (McMahan et al., 2014). Use gestalt concepts of perceptual organization (Section 20.4) when designing a panel—use position, color, and shape to emphasize relationships between widgets. For example, put widgets with similar functions close together.

Limitations

Widgets and panels are not as obvious or intuitive as direct mappings and may take longer to learn. The placement of panels and widgets have a big impact on usability. If panels are not within personal space or attached to an appropriate reference frame (Section 26.3), then the widgets can be difficult to use. If the widgets are too high, gorilla arm (Section 14.1) can result for actions that take longer than a few seconds. If the widgets are in front of the body or head, then they can be annoying due to occluding the view (although this can be reduced by making the panel translucent). If

Meta Platforms, Inc.
Exhibit 1067
Page 364

not oriented to face the user, information on the widget can be difficult to see. Large panels can also block the user's view.

Exemplar Interaction Techniques

2D desktop integration. Many panels are straightforward adaptations of windows from 2D desktop systems (Figure 28.8, left). An advantage of using desktop metaphors is their familiar interaction style and thus users have an instant understanding of how to use them. **2D desktop integration** brings existing 2D desktop applications into the environment via texture maps and mouse control via pointing. The system shown in Figure 3.5 also provides desktop metaphors such as double-clicking on a selected window header to minimize the window to a small cube (or double clicking the cube to make it large again). Bringing in such existing WIMP (Windows Icon Mouse Pointer) applications is typically not ideal from a 3D interaction perspective, but doing so does provide access to software that otherwise could only be accessed by exiting the virtual world. For example, an existing 2D calculator app could be available as a cube attached to the user's "belt." The user then simply selects and double clicks the cube to use the calculator tool without causing a break-in-presence.

Ring menus. A **ring menu** is a rotary 1D menu where a number of options are displayed concentrically about a center point (Liang and Green, 1994; Shaw and Green, 1994). Options are selected by rotating the wrist until the intended option rotates into the center position or a pointer rotates to the intended item (Figure 28.8, center). Ring menus can be useful but can cause wrist discomfort when large rotations are required. Non-isomorphic rotations (Section 28.2.1) can be used to make small wrist rotations map to a larger menu rotation.

Pie menus. **Pie menus** (also known as marking menus) are circular menus with slice-shaped menu entries, where selection is based on direction, not distance. A disadvantage of pie menus is they take up more space than traditional menus (although using icons instead of text can help) and muscle-memory precision is limited to a small number of items. Advantages of pie menus compared to traditional menus are that they are faster, more reliable with less error, and have equal distance for each option (Callahan et al., 1988). The most important advantage, however, may be that commonly used options are embedded into muscle memory as usage increases. That is, pie menus are self-revealing gestures by showing users what they can do and directing how to do it. This helps novice users become experts in making directional gestures. For example, if a desired pie menu option is in the lower-right quadrant, then the user learns to initiate the pie menu, and then move the hand to the lower

Meta Platforms, Inc.
Exhibit 1067
Page 365

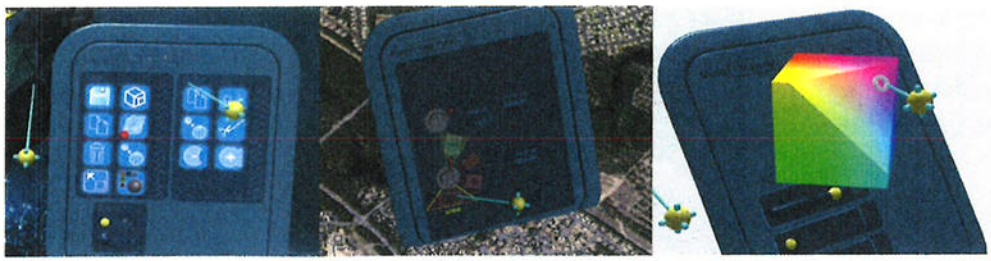


Figure 28.8 **Three examples of hand-held panels with various widgets. The left panel contains standard icons and radio buttons. The center panel contains buttons, a dial, and a rotary menu that can be used as both a ring menu and a pie menu. The right image contains a color cube that the user is selecting a color from.** (Images courtesy of Sixense and Digital ArtForms)

right. After extended usage, users can perform the task without the need to look at the pie menu or even for it to be visible. Some pie menu systems only display the pie menu after a delay so the menu does not show up and occlude the scene for fast expert users—known as "mark ahead" since the user marks the pie menu element before it appears (Kurtenbach et al., 1993).

Hierarchical pie menus can be used to extend the number of elements. For example, to change a color of an object to red, a user might (1) select the object to be modified, (2) press a button to bring up a property's pie menu, (3) move the hand to the right to select "colors," which brings up a "colors" pie menu, or (4) move the hand down to select the color red by releasing the button. If the user had to commonly change objects to the color red then she would quickly learn to simply move the hand to the right and then down after selecting the object and initiating the pie menu.

Gebhardt et al. (2013) compared different VR pie menu selection methods and found pointing to take less time and to be more preferred by users compared to hand projection (i.e., translation) or wrist roll (twist) rotation.

Color cube. A **color cube** is a 3D space that users can select colors from. Figure 28.8 (right) shows a 3D color cube widget—the color selection puck can be moved with the hand in 2D about the planar surface while the planar surface can be moved in and out.

Finger menus. **Finger menus** consist of menu options attached to the fingers. A pinch gesture with the thumb touching a finger can be used to select different options. Once learned the user is not required to look at the menus; the thumb simply touches the

Meta Platforms, Inc.
Exhibit 1067
Page 366

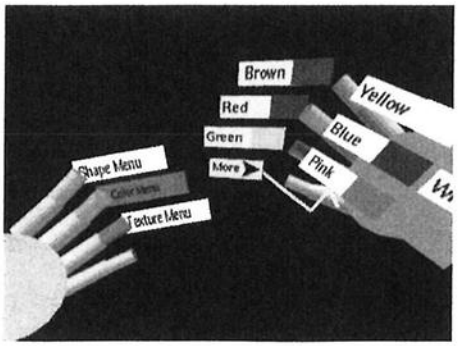


Figure 28.9 **TULIP menus.** (From Bowman and Wingrave (2001))

appropriate finger. This prevents occlusion as well as decreases fatigue. The non-dominant hand can select a menu (up to four menus) and the dominant hand can then select one of four items within that menu.

For complex applications where more options are required, a TULIP menu (Three-Up, Labels In Palm) (Bowman and Wingrave, 2001) can be used. The dominant hand contains three menu options at a time and the pinky contains a "more" option. When the "more" option is selected, then the other three finger options are replaced with new options. By placing the upcoming options on the palm of the hand, users know what options will become available if they select the "more" option (Figure 28.9).

Above-the-head widgets and panels. **Above-the-head widgets and panels** are placed out of the way above the user, and accessed via reaching up and pulling down the widget or panel with the non-dominant hand. Once the panel is released, then it moves up to its former location out of view. The panel might be visible above, especially for new users, but after learning where the panels are located relative to the body, the panel might be made invisible since the user can use his sense of proprioception to know where the panels are without looking. Miné et al. (1997) found users could easily select among three options above their field of view (up to the left, up in the middle, and up to the right).

Virtual hand-held panels. If a widget or panel is attached somewhere in the environment, then it can be difficult to find. If it is locked in screen space, then it can occlude the scene. One solution is to use **virtual hand-held panels** that have the advantage of always being available (as well as turned off) at the click of a button. Attaching the panel to the hand greatly diminishes many of the problems of world-spaced panels (e.g., panels that are difficult to read or get in the way of other objects can be reori-

Meta Platforms, Inc.
Exhibit 1067
Page 367

ented and moved in an intuitive way without any cognitive effort). The panel should be attached to the non-dominant hand, and the panel is typically interacted with by pointing with the dominant hand (Figure 28.8). Such an interface provides a "double dexterity" where the panel can be brought to the pointer and the pointer can be brought to the panel.

Physical panels. Virtual panels offer no physical feedback, which can make it difficult to make precise movements. A **physical panel** is a real-world tracked surface that the user carries and interacts with via a tracked finger, object, or stylus. Using a physical panel can provide fast and accurate manipulation of widgets (Stoakley et al., 1995; Lindeman et al., 1999) due to the surface acting as a physical constraint when touched. The disadvantage of a physical panel is that users can become fatigued from carrying it and it can be misplaced if set down. Providing a physical table or other location to set the panel on can help reduce this problem where the panel still travels with the user when virtually moving. Another option is to strap the panel to the forearm (Wang and Lindeman, 2014). Alternatively, the surface of the arm and/or hand can be used in place of a carried panel.

28.4.2 Non-spatial Control Pattern

Related Patterns

Multimodal Pattern (Section 28.5.3).

Description

The **Non-spatial Control Pattern** provides global action performed through description instead of a spatial relationship. This pattern is most commonly implemented through speech or gestures (Section 26.4).

When to Use

Use when options can be visually presented (e.g., gesture icons or text to speak) and appropriate feedback can be provided. This pattern is best used when there are a small number of options to choose from and when a button is available to push-to-talk or push-to-gesture. Use voice when moving the hands or the head would interrupt a task.

Limitations

Gestures and accents are highly variable from user to user and even for a single user. There is often a trade-off of accuracy and generality—the more gestures or words to be recognized then the less accurate the recognition rate (Section 26.4.2). Defining each gesture or word to be based on its invariant properties and to be independent from

Meta Platforms, Inc.
Exhibit 1067
Page 368

others makes the task easier for both the system and the user. System recognition of voice can be problematic when many users are present or there is a lot of noise. For important commands, verification may be required and can be annoying to users.

Even for hypothetically perfectly working systems, providing too many options for users can be overwhelming and confusing. Typically, it is best to only recognize a few options that are simple and easy to remember.

Depending too heavily on gestures can cause fatigue, especially if the gestures must be made often and above the waist. Some locations are inappropriate for speaking (e.g., a library) and some users are uncomfortable speaking to a computer.

Exemplar Interaction Techniques

Voice menu hierarchies. **Voice menu hierarchies** (Darken, 1994) are similar to traditional desktop menus where submenus are brought up after higher-level menu options are selected. Menu options should be visually shown to users so users explicitly know what options are available. See Section 26.4.2 for more information about speech recognition.

Gestures. Gestures (Section 26.4.1) can work well for non-spatial commands. Gestures should be intuitive and easy to remember. For example, a thumbs-up to confirm, raising the index finger to select "option 1," or raising the index and middle finger to select "option 2." Visual signifiers showing the gesture options available should be presented to the user, especially for users who are learning the gestures. The system should always provide feedback to the user when a gesture has been recognized.

28.5 Compound Patterns

Compound Patterns combines two or more patterns into more complicated patterns. **Compound Patterns** include the Pointing Hand Pattern, World-in-Miniature Pattern, and Multimodal Pattern.

28.5.1 Pointing Hand Pattern

Related Patterns

Pointing Pattern (Section 28.1.2), Direct Hand Manipulation Pattern (Section 28.2.1), and Proxy Pattern (Section 28.2.2).

Description

Hand selection (Section 28.1.1) has limited reach. Pointing (Section 28.1.2) can be used to select distant objects and does not require as much hand movement. However, pointing is often (depending on the task) not good for spatially manipulating objects

Meta Platforms, Inc.
Exhibit 1067
Page 369

because of the radial nature of pointing (i.e., positioning is done primarily by rotation about an arc around the user) (Bowman et al., 2004). Thus, pointing is often better for selection and a virtual hand is better for manipulation. The **Pointing Hand Pattern** combines the Pointing and Direct Hand Manipulation Patterns together so that far objects are first selected via pointing and then manipulated as if held in the hand. The user's real hand can also be thought of (and possibly rendered) as a proxy to the remote object.

When to Use

Use the Pointing Hand Pattern when objects are beyond the user's personal space.

Limitations

This pattern is typically not appropriate for applications requiring high interaction fidelity.

Exemplar Interaction Techniques

HOMER. The **HOMER technique** (Hand-centered Object Manipulation Extending Ray-casting) (Bowman and Hodges, 1997) causes the hand to jump to the object after selection by pointing, enabling the user to directly position and rotate the object as if it were held in the hand. The scaled HOMER technique (Wilkes and Bowman, 2008) scales object movement based on how fast the hand is moving (i.e., object translation is based on hand velocity). Fast hand motions enable gross manipulation whereas slow hand motions enable more precise manipulation, providing flexibility of object placement.

Extender grab. The **extender grab** (Miné et al., 1997) maps the object orientation to the user's hand orientation. Translations are scaled depending on the distance of the object from the user at the start of the grab (the further the object, the larger the scale factor).

Scaled world grab. A **scaled world grab** scales the user to be larger or the environment to be smaller so that the virtual hand, which was originally far from the selected object, can manipulate the object in personal space (Miné et al., 1997). Because the scaling is about the midpoint between the eyes, the user often does not realize scaling has taken place. If the interpupillary distance is scaled in the same way, then stereoscopic cues will remain the same. Likewise, if the virtual hand is scaled appropriately, then the hand will not appear to change size. What is noticeable is head-motion parallax due to the same physical head movement mapping to a larger movement relative to the scaled-down environment.

Meta Platforms, Inc.
Exhibit 1067
Page 370

28.5.2 World-in-Miniature Pattern

Related Patterns

Image-Plane Selection Pattern (Section 28.1.4), Proxy Pattern (Section 28.2.2), 3D Multi-Touch Pattern (Section 28.3.3), and Automated Pattern (Section 28.3.4).

Description

A **world-in-miniature** (WIM) is an interactive live 3D map—an exocentric miniature graphical representation of the virtual environment one is simultaneously immersed in (Stoakley et al., 1995). An avatar or "doll" representing the self matches the user's movements giving an exocentric view of oneself in the world. A transparent viewing frustum extending out from the doll's head showing the direction the user is looking along with the user's field of view is also useful. When the user moves his smaller avatar, he also moves in the virtual environment. When the user moves a proxy object in the WIM, the object also moves in the surrounding virtual environment. Multiple WIMs can be created to view the world from different perspectives, the VR equivalent of 3D CAD Windowing System.

When to Use

WIMs work well as a way to provide situational awareness via an exocentric view of oneself and the surrounding environment. WIMs are also useful to quickly define user-defined proxies and to quickly move oneself.

Limitations

A straightforward implementation can cause confusion due to the focus being on the WIM, not on the full-scale virtual world.

A challenge with rotating the WIM is that it results in a lack of directional compliance (Section 25.2.5); translating and rotating a proxy within the WIM can be confusing when looking at the WIM and larger surrounding world from different angles. Because of this challenge, the orientation of the WIM might be directly linked to the larger world orientation to prevent confusion (a forward-up map; Section 22.1.1). However, this comes at the price of not being able to lock in a specific perspective (e.g., to keep an eye on a bridge as the user is performing a task at some other location) as the user reorients himself in the larger space. For advanced users, there should be an option to turn on/off the forward-up locked perspective.

Exemplar Interaction Techniques

Voodoo dolls. Voodoo dolls use image-plane selection techniques (Section 28.1.3) by having users temporarily create hand-held proxies of distant objects called dolls (Pierce et al., 1999). Dolls are selected and used in pairs (with each doll representing

Meta Platforms, Inc.
Exhibit 1067
Page 371

a different object or set of objects in the scene). The doll in the non-dominant hand (typically representing multiple objects as a partial WIM) acts as a frame of reference, and the doll in the dominant hand (typically representing a single object) defines the position and orientation of its corresponding world object relative to the frame-of-reference doll. This provides the capability for users to quickly and easily position and orient objects relative to each other in the larger world. For example, a user can position a lamp on a table by first selecting the table with the non-dominant hand and then selecting the lamp with the dominant hand. Now, the user simply places the lamp doll on top of the table doll. The table does not move in the larger virtual world and the lamp is placed relative to that table.

Moving into one's own avatar. By moving one's own avatar in the WIM, the user's egocentric viewpoint can be changed (i.e., a proxy controlling one's own viewpoint). However, directly mapping the doll's orientation to the viewpoint can cause significant motion sickness and disorientation. To reduce sickness and confusion, the doll pose can be independent of the egocentric viewpoint; then when the doll icon is released or the user gives a command, the system automatically animates/navigates or teleports the user into the WIM via an automated viewpoint control technique (Section 28.3.4) where the user "becomes" the doll (Stoakley et al., 1995). This avoids the problem of shifting the user's cognitive focus back and forth from the map to the full-scale environment. That is, users think in terms of either looking at their doll in an exocentric manner or looking at the larger surrounding world in an egocentric manner, but not both simultaneously. In practice, users do not perceive a change in scale of either themselves or the WIM; they express a sense of going to the new location. Using multiple WIMs allows each WIM to act as a portal to a different, distant space.

Viewbox. The viewbox (Mlyniec et al., 2011) is a WIM that also uses the Volume-Based Selection Pattern (Section 28.1.4) and 3D Multi-Touch Pattern (Section 28.3.3). After capturing a portion of the virtual world via volume-based selection, the space within that box is referenced so that it acts as a real-time copy. The viewbox can then be selected and manipulated like any other object. The viewbox can be attached to the hand, head (so it is always in the user's view), or body (e.g., near the torso acting as a belt tool). In addition, the user can reach inside the space and manipulate that space via 3D Multi-Touch (e.g., translate, rotate, or scale) in the same way that she can manipulate the larger surrounding space. Because the viewbox is a reference, any object manipulation that occurs in either space occurs in the other space (e.g., painting on a surface in one space results in painting in both spaces). Note due to the object being a reference, care must be taken to stop the recursive nature of the view, otherwise an infinite number of viewboxes within other viewboxes can result.

Meta Platforms, Inc.
Exhibit 1067
Page 372

28.5.3 Multimodal Pattern

Description

Multimodal Patterns integrate different sensory/motor input modalities together (e.g., speech with pointing). Section 26.6 categorizes how different modalities can work together for interaction.

When to Use

Multimodal Patterns are appropriate when multiple facets of a task are required, reduction in input error is needed, or no single input modality can convey what is needed.

Limitations

Multimodal techniques can be difficult to implement due to the need to integrate multiple systems/technologies into a single coherent interface and the risk of multiple points of failure. Techniques can also be confusing to users if not implemented well. In order to keep interaction as simple as possible, a multimodal technique should not be used unless there is a good reason to do so.

Exemplar Interaction Techniques

Immersive "put-that-there." The classic example of the Multimodal Pattern is a "put-that-there" interface (Bolt, 1980) that uses a combination of the Pointing Pattern (to select the "that" and "there") and the Non-spatial Control Pattern via voice (to select the verb "put"). Neely et al. (2004) implemented an immersive form of a "put-that-there" style of region definition. With this technique, the user names and defines the vertices of a polygonal region on a terrain via pointing and speaking. An example is "Make Target Zone Apple from here (pointing gesture) to here (pointing gesture) . . . and to here (pointing gesture)." Figure 32.2 shows the architectural block diagram for this system.

For manipulating a single object, a "that-moves-there" interface (Section 26.6) where the object is selected before specifying the action is often more efficient (Section 26.5).

Automatic mode switching. One knows when the hand is within the center of vision and can effortlessly bring the hand into view when needed. Mode switching can take advantage of this. For example, an application might switch to an image-plane selection technique (Section 28.1.3) when the hand is moved to the center of vision and to a pointing selection technique (Section 28.1.2) when moved to the periphery (since the source of a ray pointer does not need to be seen).

Meta Platforms, Inc.
Exhibit 1067
Page 373

Interaction: Design Guidelines

Just as there is no single tool in the real world that solves every problem, there is no single input device, concept, interaction pattern, or interaction technique that is best for all VR projects. Although it is generally better to use the same interaction metaphor across different types of tasks, that is not always possible or appropriate. Interaction designers should take into account the specific task when choosing, modifying, or creating new interactions.

Human-Centered Interaction (Chapter 25)

Intuitiveness (Section 25.1)

- Focus on making interfaces intuitive—that is, design interfaces that can quickly be understood, accurately predicted, and easily used.
- Use interaction metaphors (concepts that exploit specific knowledge that users already have of other domains) to help users quickly develop a mental model of how an interface works.
- Include within the virtual world everything that is needed to help users form a consistent conceptual model of how things work (e.g., within world tutorials). Users should not have to rely on any external explanation.

Norman's Principles of Interaction Design (Section 25.2)

- Practice human-centered design and follow well-known general principles to help users create simplified mental models of how the interactions work. These include consistent affordances, unambiguous signifiers, constraints to guide actions and ease interpretation, obvious and understandable mappings, and immediate and useful feedback.

Meta Platforms, Inc.
Exhibit 1067
Page 374

Affordances (Section 25.2.1)

- Remember that an affordance is not a property of an object, but is a relationship between an object and a user. Affordances don't only depend on the object being afforded, but also depend on the user; affordances may be different for users with different capabilities.

Signifiers (Section 25.2.2)

- Make affordance perceivable through signifiers. A good signifier informs users what is possible before interacting.
- Make obvious to the user the state of the current interaction mode.

Constraints (Section 25.2.3)

- Where appropriate, add constraints to limit possible actions and to improve accuracy and efficiency.
- Use constraints to add realism (e.g., do not allow users to travel through walls).
- Do not assume real-world rules such as gravity are always appropriate. For example, hanging tools in the space around the user makes them easier to grab.
- Use signifiers wisely to prevent users from making the wrong assumptions about constraints. If a user doesn't know what to do, she is effectively constrained.
- Make constraints consistent so learning can be transferred across tasks.
- Consider allowing expert users to remove constraints for advanced interactions.

Feedback (Section 25.2.4)

- Use feedback substitution when haptics are not available. For example, use audio and highlighting to signify touching of objects.
- Do not overwhelm users with too much feedback.
- Consider putting information in front of the user in the torso reference frame near the waist rather than on a heads-up display in the head reference frame.
- If a heads-up display in the head reference frame must be used, only present minimal information on the display.
- Provide the capability to turn on/off (or make visible/invisible) widgets, tools, and interface cues.

Meta Platforms, Inc.
Exhibit 1067
Page 375

Mappings (Section 25.2.5)

- To maximize performance and satisfaction, maintain directional compliance, nulling compliance, and temporal compliance.
- Focus first on creating mappings that have directional compliance (the direction of sensory feedback should match the direction of the interface device) so users can anticipate motion in response to their physical input.
- For fully direct interactions, maintain position compliance (i.e., make the virtual position of objects match the physical position of the device).
- If position compliance is not appropriate, use nulling compliance (the virtual object should return to its original location when the device returns to its original location).
- Use nulling compliance to take advantage of muscle memory.
- Choose absolute input devices over relative input devices in order to maintain compliance.
- If the results of an interaction cannot be immediately computed (i.e., a lack of temporal compliance), then provide some form of immediate feedback to inform the user the problem is being worked on.
- For non-spatial mappings, use commonly accepted metaphors (e.g., up is more, down is less).

Direct vs. Indirect Interaction (Section 25.3)

- Use tools to extend one's reach so users feel like they are directly interacting.
- Use direct, semi-direct, and indirect interactions where appropriate and not where not appropriate. Do not try to force everything to be direct.

The Cycle of Interaction (Section 25.4)

- To design or improve interactions, use Norman's seven stages of interaction to break up typically subconscious processes into explicit steps.
- Think about what stages are missing or what stages do not work well that make interaction difficult. Then add or modify signifiers, constraints, mappings, and feedback as appropriate for each stage.
- Use the seven stages of interaction with task analysis as a stepping stone toward implementation.

Meta Platforms, Inc.
Exhibit 1067
Page 376

The Human Hands (Section 25.5)

- Support interactions with two hands where appropriate.
- Do not assume just because two hands are used for an interface that the interface will be better. Two-handed interfaces can be difficult to use if designed inappropriately.
- Feedback from users is even more essential for two-handed interaction than one-handed interaction.
- Have the non-dominant hand maintain the reference frame so the dominant hand can work precisely without having to work in a locked position.
- Design bimanual interactions to work together in a fluid manner, switching between symmetric and asymmetric modes as appropriate for the task.

VR Interaction Concepts (Chapter 26)

Interaction Fidelity (Section 26.1)

- Consider using realistic interactions for training applications, simulations, surgical applications, therapy, and human-factors evaluations.
- Consider using non-realistic interactions for increasing performance and minimizing fatigue.
- Use magical interactions to enhance the user experience, circumvent the limitations of the real world, and teach abstract concepts.
- Consider interaction metaphors as a source of inspiration. Unless realistic interaction is a primary goal, use intuitive and useful magical techniques.

Proprioceptive and Egocentric Interaction (Section 26.2)

- Exploit the one real object every user has—the human body.
- Use torso tracking in addition to head and hands tracking whenever possible to maximize proprioception. Alternatively, chair rotation can be tracked to estimate torso rotation.
- Place commonly used tools relative to the body to take advantage of muscle memory so users do not have to take visual attention away from that which is being worked on (or so tools do not even have to be within the user's field of view).

Meta Platforms, Inc.
Exhibit 1067
Page 377

- Don't assume exocentric perspectives preclude egocentric perspectives. Take advantage of egocentric intuition even when designing an exocentric experience.

Reference Frames (Section 26.3)

- Provide the capability for users to think and interact in an exocentric virtual-world reference frame when the intent is for users to create content over a wide area, form a cognitive map of the environment, determine their own global position, or plan travel on a large scale.
- Be careful of placing direct interfaces in virtual-world reference frames as they can be awkward to reach without easy and precise navigation capability.
- Use real-world reference frames as rest frames (e.g., an automobile interior, a cockpit, or non-realistic cues) for users to feel stabilized in space and to reduce motion sickness.
- Provide a place for users to set physical devices when not needed and render those objects in the real-world reference frame so they can easily be seen and picked up.
- Place information, interfaces, and tools relative to the body by placing them in the torso reference frame.
- Allow advanced users to turn on/off or make invisible (but still usable) items in the torso, hand, and head reference frames.
- Place a visual representation of hand controllers in the torso reference frame for non-tracked controllers and in the hand reference frame for tracked hand-held controllers.
- Place signifiers in the hand reference frame pointing to buttons, analog sticks, and/or fingers so it is obvious what they do. Provide the capability for users to turn them on and off.
- Minimize cues in the head reference frame for anything other than a pointer if head tracking is used for input.

Speech and Gestures (Section 26.4)

- Use explicit visual signifiers (e.g., a list of words or hand icons) to portray what speech or gesture commands are available. Provide feedback by highlighting the selected option.
- Have users verify important commands to prevent major errors.

Meta Platforms, Inc.
Exhibit 1067
Page 378

- Use a small number of words or gestures that are well defined, natural, easy for the user to remember, and easy for the system to recognize.
- When more than one person will be present in the same space, use push-to-talk and/or push-to-gesture to prevent accidental unintended commands.
- Use direct gestures for immediate system response.
- When users have their own unique system, use speaker-dependent recognition if they are willing to train the system and adaptive recognition if they are not.
- Reduce the potential for error by only allowing a subset of vocabulary to be recognized depending upon the context.

Modes and Flow (Section 26.5)

- When multiple interaction modes are used, make it clear to the user what the current mode is.
- Use object-action (or selection-manipulation) sequences over action-object sequences.
- Enable smooth and easy transition between selecting an object and manipulating or using that object.
- Minimize distractions to enhance the flow of interactions and allow for full attention to the primary task.
- Design interactions so users do not have to physically (whether the eyes, head, or hands) or cognitively move between tasks.
- Use lightweight mode switching, physical props, and multimodal techniques to help maintain the flow of interaction.

Multimodal Interaction (Section 26.6)

- Use a single specialized input modality when it is clear that single modality is best for the task and there is no reason to include other modalities. Do not add a modality just for the sake of adding a modality.
- Consider using equivalent input modalities when user preferences are strongly divided.
- Use redundant input modalities to reduce noise and ambiguous signals.
- Use concurrent input modalities to improve efficiency by enabling the user to perform two interactions simultaneously.
- Use complementarity input modalities for "put-that-there" or "that-moves-there" types of interfaces.

Meta Platforms, Inc.
Exhibit 1067
Page 379

- Use transfer input modalities when one modality device is unreliable so users will not have to start over if there is a failure.

Beware of Sickness and Fatigue (Section 26.7)

- Be extra careful of viewpoint control techniques that can induce motion sickness. Study Part III and follow the guidelines in Chapter 19 to minimize adverse health effects.
- When motion sickness is a primary concern (e.g., for users new to VR or a general audience), only allow one-to-one mapping of real head motion or use teleportation.
- Avoid using and creating interactions that require the user to hold the hands up high or in front of the body for more than a few seconds at a time. Use devices that do not require line of sight so interactions can be performed comfortably in the lap or from the sides of the body.

Visual-Physical Conflict and Sensory Substitution (Section 26.8)

- Enforce physics constraints when hand penetration into objects is shallow so the hand does not pass through the object surface. Do not enforce physics constraints when penetration is deep; instead allow the hand to pass through virtual objects.
- Consider drawing two hands for a single physical hand—one that penetrates objects and one that doesn't.
- Use highlighting to convey an object is selectable when the hand is close to the object.
- Use audio to convey collisions.
- Use passive haptics or vibrotactile haptics whenever possible.
- If training transfer is not important, use ghosting to signify a new potential position until the user confirms placement.

Input Devices (Chapter 27)

Input Device Characteristics (Section 27.1)

- Match the interaction technique with the device and match the device with the interaction technique. Understand the different device characteristics and classes in order to determine what is best for the project.

Meta Platforms, Inc.
Exhibit 1067
Page 380

- Use 6 DoF devices whenever possible and then decrease DoFs in software where appropriate.
- Choose input devices that work in a user's entire personal space, do not require line of sight, are robust to various lighting conditions, and work for all hand orientations.
- Use buttons for tasks that are binary, when the action needs to occur often, when immediate response and reliability are required, when abstract interactions are appropriate, and when physical feedback of feeling the button pressed/released is important.
- Do not overwhelm users with too many buttons.
- Use bare-hand systems, gloves, and/or haptic devices that correspond to virtual objects when a high sense of realism and presence is important.

Classes of Hand Input Devices (Section 27.2)

- If not sure what to use or there is no strong preference, then start with tracked hand-held controllers. They are currently the best option for a majority of interactive VR experiences.
- For public location-based entertainment, consider building customized interfaces such as world-grounded devices that are optimized for the particular experience.
- Attach labels to virtual representations of controllers to signify what the controls do.
- Use tracked hand-held controllers when the user commonly holds virtual devices to enhance presence through physical touch.
- Don't assume gloves are only for full hand tracking. Consider using pinch gloves that don't require holding a physical controller but have the advantages of buttons via pressing the fingers together.

Classes of Non-hand Input Devices (Section 27.3)

- If eye tracking is available, do not overuse it. Instead use eye gaze for specialized tasks and in subtle ways (e.g., have virtual characters respond when looked at).
- When designing eye gaze interactions, maintain the natural function of the eyes, augment rather than replace, focus on interaction design, improve the

Meta Platforms, Inc.
Exhibit 1067
Page 381

interpretation of eye movements, choose appropriate tasks, use passive gaze over active gaze, and leverage gaze information for other interactions.

- Use microphones that are specially designed for speech recognition.

Interaction Patterns and Techniques (Chapter 28)

Selection Patterns (Section 28.1)

- When interactions do not need to be realistic, use the Pointing Pattern or the Image-Plane Selection Pattern.

Hand Selection Pattern (Section 28.1.1)

- Use the Hand Selection Pattern when interactions need to be realistic.
- For high interaction fidelity, use a realistic virtual hand to select objects.
- For mid-interaction fidelity, use hands without arms to reach just beyond personal space (i.e., near action space).
- Consider the go-go technique for personal space and midrange action space.

Pointing Pattern (Section 28.1.2)

- For more controlled pointing, consider using a precision mode.
- For selecting objects that are small, consider using pointing with object snapping.
- Do not use head pointing unless head tracking is the only tracking available or there is another good reason to do so.
- Do not use eye tracking for selecting objects unless there is a good reason to do so.

Image-Plane Selection Pattern (Section 28.1.3)

- For easy-to-use touch at a distance, use image-plane selection. However, do not use when the user must frequently select objects as gorilla arm can result.

Volume-Based Selection Pattern (Section 28.1.4)

- Use the Volume-Based Selection Pattern when selecting data/space that has no geometric surfaces.
- Be careful of requiring volume-based selection for novice users.

Meta Platforms, Inc.
Exhibit 1067
Page 382

Manipulation Patterns (Section 28.2)

Direct Hand Manipulation Pattern (Section 28.2.1)

- Use the Direct Hand Manipulation Pattern unless there is a reason not to as it is more efficient and satisfying than other manipulation patterns.
- For high interaction fidelity, use a virtual hand for both selection and manipulation.
- Consider using non-isomorphic rotations to reduce clutching and to increase performance and precision.

Proxy Pattern (Section 28.2.2)

- Use the Proxy Pattern to intuitively manipulate remote objects or when the user scales himself or the world.
- Use tracked physical props for direct action-task correspondence. Such props result in an easy-to-use interface that does not require any training, facilitate natural two-handed interactions, and provide tactile feedback to the user.

3D Tool Pattern (Section 28.2.3)

- Use the 3D Tool Pattern to enhance the capability of the hands to manipulate objects.
- Use signifiers for object-attached tools to make it obvious how to use those tools to manipulate the object.
- To enable precise modeling and reduce complexity, use jigs that enable user-controlled constraints, snapping, and discrete manipulations.

Viewpoint Control Patterns (Section 28.3)

- Be especially careful of sickness and injury when choosing, designing, and implementing viewpoint control techniques, especially for users new to VR. See Part III.

Walking Pattern (Section 28.3.1)

- Use the Walking Pattern when high biomechanical symmetry and high presence is desired, fatigue is not a concern, and safety measures to prevent tripping, physical collisions, and falling are covered.
- Use real walking when the physically tracked space is as large as or larger than the virtual walkable space, and both spatial understanding and minimizing sickness are important.

Meta Platforms, Inc.
Exhibit 1067
Page 383

- Use redirected walking when the physically tracked space is smaller than the virtual walkable space.
- Use walking in place when the physically tracked space is small or safety is a concern.
- Use a treadmill for running and when walking/running vast distances is required.

Steering Pattern (Section 28.3.2)

- Use the Steering Pattern when sickness is not a primary concern, interaction fidelity is not important, acceleration/deceleration can be minimized, and real-world stabilized cues can be provided.
- Make steering as simple as possible to minimize cognitive load so the user can focus on spatial knowledge acquisition and information gathering.
- Provide visual cues to help users know what direction is forward.
- Consider not using virtual rotations if torso/chair tracking is available and the system is wireless.
- If virtual rotations are required and the user can be constrained to the ground, use dual analog sticks.

3D Multi-Touch Pattern (Section 28.3.3)

- For applications not requiring high interaction fidelity, use the 3D Multi-Touch Pattern when creating assets, manipulating abstract data, viewing scientific datasets, or rapidly exploring large and small areas of interest from arbitrary viewpoints.
- Consider adding constraints (e.g., force uprightness, limit scale, and/or disable rotations) for novice users.
- Provide a visual indication of the center of rotation/scale.

Automated Pattern (Section 28.3.4)

- Use the Automated Pattern when free exploration of the environment is not desired or not possible.
- Use teleportation when traveling large distances, between worlds, when efficiency is important, or when motion sickness must be minimized. Do not use if spatial orientation must be maintained.
- Use smooth transitions when maintaining spatial orientation is the primary concern.

Meta Platforms, Inc.
Exhibit 1067
Page 384

- When passively moving the user, reduce motion sickness by keeping visual velocity as constant as possible, providing stable real-world reference cues (e.g., a cockpit), and/or providing a leading indicator.

Indirect Control Patterns (Section 28.4)

- Use an Indirect Control Pattern when spatial mappings are not appropriate or details of how something is done are not important to the user. Uses include controlling the overall system, issuing commands, changing modes, and changing non-spatial parameters.
- For indirect control, make signifiers obvious, such as the shape and size of controls, their visual representation and labeling, and apparent affordances of their underlying control structure.

Widgets and Panels Pattern (Section 28.4.1)

- Use the Widgets and Panels Pattern when it is difficult to directly interact with an object.
- Where appropriate, use well-known 2D interaction metaphors such as pull-down/pop-up menus, radio buttons, and checkboxes.
- Use gestalt concepts of perceptual organization when designing a panel—use position, color, and shape to emphasize relationships between widgets. For example, put widgets with similar functions close together.
- Place widgets and panels in a way that is easy for users to access (e.g., on the non-dominant hand or in the torso reference frame).
- For commonly used commands, use pie/marking menus to teach gestures and to embed those gestures into muscle memory.
- For pie/marking menus, use pointing over projection or roll.
- If pinch gestures are available, place menu options on the fingers.
- Consider placing panels or widgets above the head so users can pull them down when needed.
- Consider placing a panel in the non-dominant hand that can be turned on or off, and where widgets on the panel are controlled by the non-dominant hand.
- For 2D tasks that require precision, consider using a physical panel the user holds in the non-dominant hand or that is attached to the forearm.

Meta Platforms, Inc.
Exhibit 1067
Page 385

Non-spatial Control Pattern (Section 28.4.2)

- Use the Non-spatial Control Pattern for global action performed through description instead of a spatial relationship.
- Use real-world words and gestures that are intuitive and easy to remember. Keep the number of options small and simple.
- Provide signifiers (e.g., icons for gestures or a list of words for voice) to remind non-expert users what options are available.
- Always provide some form of feedback.
- When accuracy is more important than speed, verify commands. Make the confirmation process fast and do not require precision. Examples include clicking a physical button or saying "confirm."
- Use push-to-talk or push-to-gesture to prevent accidental unintended commands.
- Use voice control when moving the hands or the head would interrupt a task.
- Be careful of depending on voice recognition when multiple people are in the same physical environment or there is a lot of noise.

Compound Patterns (Section 28.5)

Pointing Hand Pattern (Section 28.5.1)

- When high interaction fidelity is not required, use the Pointing Hand Pattern to select objects at a distance but to manipulate them as if held in the hand.

World-in-Miniature Pattern (Section 28.5.2)

- Use the World-In-Miniature Pattern to provide situational awareness, to quickly define user-defined proxies, or to quickly move oneself.
- Consider using the equivalent of a forward-up map for the world-in-miniature so that the orientation of the map matches the orientation of the larger world. When appropriate, provide the capability to turn this feature off.
- To reduce sickness, do not directly map the user's doll in the world-in-miniature to the user's movement. Instead, animate/navigate or teleport the viewpoint into the doll when commanded by the user.

Multimodal Pattern (Section 28.5.3)

- Use the Multimodal Pattern when multiple facets of a task are required, when reduction in input error is needed, or when no single modality can convey what is needed.

Meta Platforms, Inc.
Exhibit 1067
Page 386

- Only use a multimodal technique when there is a good reason to do so. When used, keep interactions as simple as possible.
- Consider using automatic mode switching when each technique is only usable in specific situations and those situations are clear to the user.

Meta Platforms, Inc.
Exhibit 1067
Page 387

PART

ITERATIVE DESIGN

Design thinking takes a solution-focused approach to problem solving, working collaboratively to iterate an endless, shifting path toward perfection. It works toward product goals via specific ideation, prototyping, implementation, and learning steps to bring the appropriate solution to light.

—Jeff Gothelf and Josh Seiden (2013)

Up to this point, this book has provided a background on the mechanisms of perception, VR sickness, content creation, and interaction. However, such mechanisms are not entirely understood and very much depend on the goals and design of the VR project. Furthermore, assumptions about the real world or how we perceive and interact with it do not always hold in VR. One cannot simply look up the numbers when designing VR experiences or get the experience perfect on the first attempt. VR design is also very different than 2D desktop or mobile application design, and there is not nearly as much known or standardized about VR. VR design must be iteratively created based off of continual redesign, prototyping, and feedback from real users. In fact, the design of many VR experiences are due to "accidental discoveries" that were not initially intended (Andrew Robinson and Sigurdur Gunnarsson, personal communication, May 11, 2015). This chapter focuses on iterative design concepts to help the team move toward ideal experiences as quickly and efficiently as possible.

With new development platforms such as Unity and other tools, VR experiences can be created in a fraction of the time they could be created compared to just a few years ago. Experienced designers can now literally create a simple VR experience in hours, without any programming experience required. Some modifications can be as easy as a click of a button, and some assets and behaviors built by others can be integrated

Meta Platforms, Inc.
Exhibit 1067
Page 388

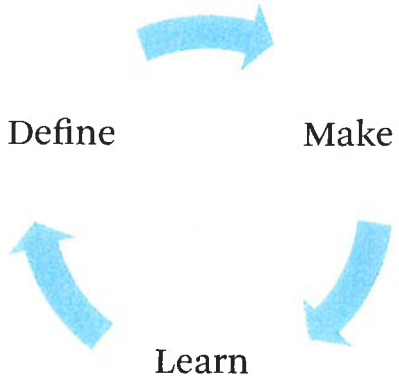


Figure VI.1 **High-level iterative process common to all VR development.**

with basic programming skills. Because of how quickly simple VR experiences can be built and modified, prototyping basic individual components can often be done in a day or less. However, to go beyond the basics, designers should either be developers themselves, learn basic programming skills for simple prototyping and testing, or work directly in pairs with developers. Note, this is not to say that professional programmers are not required for creating compelling VR experiences. Integrating advanced/innovative features and behaviors, understanding software architecture, and writing clean code takes time to do properly and are essential for creating efficient and maintainable code.

The Define-Make-Learn Cycle

Although a single overall detailed process is not appropriate for VR design, there is general agreement that iterating upon defining the project, building prototypes, and learning from users are all important. Careful consideration of these stages is critical to optimizing user experience and comfortable VR.

Part VI discusses the philosophy of iterative design and then the three iterative stages (FIgure VI.1) as summarized below.

1. **The Define Stage.** This stage attempts to answer the question "What do we make?" and includes everything from the high-level vision to listing requirements.
2. **The Make Stage.** This stage answers the question "How do we make it?" and then proceeds to make it.

Meta Platforms, Inc.
Exhibit 1067
Page 389

3. **The Learn Stage.** This stage answers the question "What works and what does not work?" The answers are fed back into the Define Stage to refine what is to be made.

Although conceptually these stages are performed in sequence, they are tightly interwoven and often occur in parallel. Within the three stages, various smaller process pieces are described that can be fit into the larger iterative process as needed. These individual pieces may or may not be used depending on the project and what part of the lifecycle the process is in. The stages are often not consistent even across iterations of the same project. Thus, it would both limit and overcomplicate the design process to try to formalize a sequence of VR design processes, or even to formalize each individual process piece. Instead, basic overviews of the process concepts are described within each of the three primary stages. The specifics will depend on the project goals and team preference.

Overview of Chapters

Part VI consists of five chapters that describe the overall iterative process as well as more detailed specific processes as they apply to creating VR experiences.

Chapter 30, Philosophy of Iterative Design, first provides a review of overarching concepts essential to VR design. Iterative design as it applies to VR is especially dependent on art, science, human-centered design, iteration, the specifics of the project, and the team creating the experience.

Chapter 31, The Define Stage, discusses how VR application ideas are created and refined. Fifteen ways of defining different aspects of the project are described in this chapter.

Chapter 32, The Make Stage, discusses the actual implementation of the idea into initial tangible prototypes and then into more polished experiences that are delivered to end users.

Chapter 33, The Learn Stage, describes core research concepts and explains how to obtain qualitative feedback from representative users, collect more objective/quantitative data, improve the design through constructivist approaches, and formally test/compare different implementations.

Chapter 34, Iterative Design: Design Guidelines, summarizes the iterative process and lists a number of actionable guidelines to help VR creators move toward engaging experiences.

Meta Platforms, Inc.
Exhibit 1067
Page 390

Philosophy of Iterative Design

Design has multiple meanings yet all are intimately related. Design is a created object, a process, and an action. Quality design focuses on utility, effectiveness, efficiency, elegance, delight, and the conveying of meaning (Brooks, 2010). For the purposes of this book, design entails the entire creation of a VR experience from information gathering and goal-setting to delivery and product enhancements/support.

30.1 VR Is Both an Art and a Science

VR is an art because we have to be creative in creating new experiences based off of common sense, rules of thumb, cultural metaphors, creative outside-the-box thinking, etc. VR is an art also because creating new worlds enables us to throw away many of the rules of the real world; VR requires innovation. Creating a quality VR experience is not just about optimizing a set of algorithms. It is more imperative for the humans behind the creation to decide on the right things to design instead for perfectly designing the wrong things. VR designers seek to find what is useful, whether realistic or magical.

VR is a science because often our intuition of what will work often doesn't work in practice. We have to get users to test our ideas, collect data, and analyze that data to help determine how to improve the experience. As in programming, iterative design consists of quickly testing many implementations, continually improving upon initial ideas. The faster we can obtain quality answers to meaningful questions, the faster we can iterate toward better and better experiences.

30.2 Human-Centered Design

The measure of progress for a VR project is the user experience. If the experience is improved then progress is being made. Traditional software development measures, such as the number of lines of code, are very different from measures that should

Meta Platforms, Inc.
Exhibit 1067
Page 391

be used for VR progress. Such measures are largely useless for VR. A well-working interaction technique that takes a couple of hours to integrate is worth far more than tens of thousands of lines of code that took months of work, but doesn't feel right. A brilliantly engineered solution is irrelevant if it doesn't work for the human.

Bad VR insists users perform abnormally, to adapt themselves to the peculiar demands of the system. When users are not able to meet the inhuman expectations of the virtual environment, then there is a design problem. Human-centered design errors are often only discovered when those external to the project experience the creation. Obtaining feedback from real users is critical early in the project, for if human-centered design error is discovered near ship time then the entire project may be deemed a failure instead of many small failures discovered and fixed in early iterations.

30.3 Continuous Discovery through Iteration

A core concept of VR design is discovery should always be occurring. Continuous discovery is the ongoing process of engaging user's during the design and development process (Gothelf and Seiden, 2013). The goal of discovery is to understand what users want to do, why they want to do it, and how they can best do it.

Knowing or rationalizing everything in advance for building complex systems results in projects becoming over-budget, overdue, and performance disasters (Brooks, 2010). This is not only because there are not-yet known answers to questions the team has at the beginning, but also because the team doesn't yet know what they don't know. Questions will lead to more questions that were not initially considered. Discovering what one did not formerly even consider is where new insights and breakthroughs in design most often occur.

Iteration is more important for developing VR systems than it is for developing non-VR systems (Wingrave and LaViola, 2010). The possibilities with VR extend beyond what is possible with real reality, yet doing one thing badly can have severe consequences. Even if everything was known about VR, then the number of factors and combinations of factors that can go into a VR experience explodes exponentially. Thus, little can be known in advance and every project will be unique.

We can't afford to wait until the end of the project to see if everything works (hint: it won't). Learning, change, and innovation are much less expensive at the beginning than at the end and there must be many options available that enable fast adjustments as necessary. The question then is not if iterative design should be used, but how to most quickly and effectively iterate. This is accomplished via rapid prototyping and getting feedback from experts and representative users as quickly and as often as

Meta Platforms, Inc.
Exhibit 1067
Page 392

possible. As advocated by the Lean Startup (Ries, 2011), fail early and fail often. Each failure teaches about what to do better. Failures are essential for exploration, creativity, and innovation. If failure is not occurring often, then one is not innovating enough and breakthroughs are not likely to occur. It is fairly easy to create a safe VR experience where there is no risk of learning. But that is also the route to a dull, uninteresting experience.

Even if everything could be known at the start of the project, change happens over time. The reality of life, business, and technical advancement dictates that what was important yesterday may not be important today. When real reality gets in the way of the plan then that reality always wins. Another challenge that is especially true for VR projects, is that when one part of the application changes, it rarely occurs in isolation. Always be on the lookout for changes in assumptions, technology, and business. Then change course as necessary through adaptive planning.

For those new to VR (this includes everyone on the team from business development to programmers), the first step of discovery is to step into the role of user and try a broad range of existing VR experiences making notes of what works and what doesn't work. Then, new insights are discovered by prototyping new or modified ideas and getting feedback from users. In fact, this is more important than planning and pre-analysis. There is more value in getting a bad prototype out than spending days thinking, considering, and debating. It is important to create a culture of accepting failure, especially early in a VR project, so that team members feel it is safe to experiment. Such experimentation encourages creativity and innovation, and leads to effective VR. Feedback and measurement from real application beats opinions based on theory.

In the end, the success or failure of the VR experience isn't the team's decision—it's the consumers of the experience. The sooner the team can discover what gets results for users, then instead of iterating in an arbitrary direction, the team can iterate toward success.

30.4 There Is No One Way—Processes are Project Dependent

In any field, no single design process is appropriate for every project. This is even truer for VR for multiple reasons. Even though VR research has occurred over multiple decades, the community is constantly learning/changing and there are not yet (and there may never be) standardized processes to follow. VR covers a broad range of industries and applications. For example, creating an assembly line automotive training system is going to have a very different process than creating an immersive film. Even within the same project there are many factors that will determine what

Meta Platforms, Inc.
Exhibit 1067
Page 393

process is most appropriate at the time. Factors include the current size of the team, approaching deadlines, and how far along the project is. Because there are so many unknowns, and large variance for what is known, what is most appropriate must be determined on a project-by-project basis, and "appropriate" will change as the project iterates toward a solution.

Given the variety of VR project demands, why define processes in advance at all? There must be some way to bring order to chaos; otherwise, a final cohesive application can't be achieved. At some point, approval must be obtained to move ahead. Schedules and budgets cannot be overrun. Customers must be kept satisfied. Processes in some form are indeed required for these things to happen. However, they should not be used blindly. Understanding various processes, and the advantages and disadvantages of each, enables a team to selectively choose from a library of processes as appropriate, and to integrate them together as needed.

Individual team members and communication between team members should have priority over sticking to a formal process, and responding to change should have priority over following a plan (Larman, 2004). This does not mean processes and plans do not matter. It does mean ongoing input from individuals, communication between the team, and appropriate change are more important. Good processes will support these things rather than hinder them.

Of course, a primary goal of processes is to help identify and prioritize matters of most importance, providing focus and clarity to the team. Good processes also provide for easy and swift exceptions—all rules can be broken when appropriate. The specifics of what processes best fit a project are best chosen based on experience. There are no rules or algorithm that state which process is most appropriate for a specific project although this chapter does provide some general guidelines. If it is not clear with what specific processes to start with, then just pick some and go with them. Just like iterating upon the project, you will iterate upon the processes, improving upon them as you use and learn from them. Do not become attached to the processes and be willing to throw them out if and when they are not working.

30.5 Teams

VR by its nature is cross-disciplinary and communication between teammates is essential for VR development. Team communication is about externalizing one's work (Gothelf and Seiden, 2013) to others through spoken conversation, sketches, whiteboards, foamcore boards, artifact walls, printouts, sticky notes, and of course prototype demos. All but spoken conversation helps to equalize team member input, regardless of whether someone is loud or quiet. It is very important to keep such forms

Meta Platforms, Inc.
Exhibit 1067
Page 394

of structural communication informal and raw as long as possible. More formal documentation should trail, not lead. Such communication should be easily modifiable with a way to easily determine what has been updated. Keep teams small to maximize communication. Individuals will likely have multiple roles. If the project is being developed as part of a larger organization, break groups into smaller teams.

A team effort often (but not always!) results in bigger and better results than an individual due to input from a range of opinions/viewpoints. However, design should not be done by committee. Collaborative design should be led by a single authoritative director (Section 20.3) that listens well but acts as a final decision maker on high-level decisions (the one exception to this is for teams of only two individuals, similar to pair programming). This director needs to be completely trusted by both the team working with him as well as by those on the business/finance side. The director should clearly understand the conceptual design of the entire project. If it is not possible for the director to understand how everything is connected then the design is overcomplicated and should be simplified.

The entire team should be on board with the overarching design philosophy. Not only should individuals be aware of what other team members are working on but also each individual should be actively involved with other's work to some degree. That is everyone should co-create, not just critique. The entire team should work toward getting the design right through prototypes and feedback from users before making final decisions on polishing what has been built. This also prevents individuals from becoming attached to their creations which results in less willingness to give up those creations.

Some team members might be subject-matter experts but have little experience with VR so they don't know what is possible or the limitation of today's technology. Other team members may be VR experts, but have little understanding of the problem domain. Those with a traditional design background should understand challenges of VR implementation. Those with a programming background should be open to learning from other team members, external suggestions from those experienced in VR, other disciplines, and user feedback. If any individual on a team is close minded to ideas that are not his own and/or other disciplines, then that individual is going to be a hindrance to creating compelling VR experiences (much more so than any other discipline due to the cross disciplinary nature of VR). Warn the team member to change his attitude, and if he is unwilling to do so then remove him from the project as soon as possible as if cancer has invaded the team (because like cancer, such attitude can kill a VR project).

Meta Platforms, Inc.
Exhibit 1067
Page 395

The Define Stage

The **Define Stage** of a project is the beginning but also continues until the end as more is discovered from the Make and Learn Stages. All parts of the Define Stage should be described from the client's or user's point of view and be able to be understood by anyone. That way multiple perspectives can be taken into account.

Be careful of "analysis paralysis" of trying to have every single detail figured out before pulling the trigger. Spending too much time at the beginning on the Define Stage can be detrimental to the project because what is defined may not be possible with today's technology or may not ultimately be what customers want. In most cases, it is best to start with only defining a general overview of the project rather than getting into too much detail. In some cases, some developers (e.g., solo indie developers) might even choose a "fire then aim" strategy of first starting with the Make Stage. In other cases, analysis and definition can be more important (e.g., command and control applications where the VR system must be integrated with existing systems and procedures). Knowing when to start implementing is largely a matter of experience and judgment of knowing when is enough. If you are not sure if you are ready to move on to the Make Stage then it is better to err on moving forward. The Define Stage will always be waiting for your return to fill in more details as necessary.

This chapter discusses several Define Stage concepts. The concepts are listed in approximate order they might occur, but like everything else in iterative design, the order is best chosen based on project needs. Not all projects will use all of the concepts, although the more concepts that are eventually used and refined, the more likely the success of the project. Note for all of these concepts, it is not just important to define *what* the project is, but it is important to state *why* those decisions were made. By doing so, the team can better justify keeping, changing, or removing each element on future iterations.

Meta Platforms, Inc.
Exhibit 1067
Page 396

31.1 The Vision

Pyramids, cathedrals, and rockets exist not because of geometry, theory of structures, or thermodynamics, but because they were first pictures—literally visions—in the minds of those who conceived them.

—Eugene Ferguson, *Engineering and the Mind's Eye* (1994)

Like any great engineering achievement, a VR project must be started by someone somewhere. What is the overall vision? What is trying to be achieved? Why is it being done? This is the chance to make it all up. With unlimited resources, what could be achieved?

The early vision is largely speculation. There is no way to know the final solution or even what all the concerns are until the project is completed. How can project be defined when there are so many unknowns? Guess! Articulated guesses beat unspoken or vague models (Brooks, 2010). Guessing forces the team to think about the exposed assumptions and to think very carefully about the design. Guessing is the beginning of an iterative journey that will better define goals as more is learned. Both the formulation of the problem and its solution will evolve together, with interchange of information between the two (Dorst and Cross, 2001).

Intelligent and informed speculation is based on the real world and its people. Refined speculation begins by talking with others as discussed in the next section.

31.2 Questions

At first thought, it seems having such creative freedom to make it all up would be easy to do. However, this is often the stage of the project that can be the most challenging. In fact, Brooks (2010) states "the hardest part of design is deciding what to design." Fortunately VR projects do not exist in isolation. Understanding the background and context of how the project will fit in with the larger picture helps to define the project.

The most important time to get feedback from other is during the conceptualization of the idea. Innovative ideas, ones that create new interactions, experiences, and significantly better results come about by reconsidering the goals and focusing on what people really want. The best way to determine what people really want is accomplished by talking with people and asking lots of questions. However, it's not as simple as asking what they want. Often, what people think they want is not what they actually want. As a consultant, the author has found one of the greatest services to provide is to help clients discover what they want. Don't expect people to straight out tell you what they want and do not expect your assumptions of what they want to be correct. Asking the right questions is essential.

Meta Platforms, Inc.
Exhibit 1067
Page 397

The most important questions to ask at this stage begin with the word "why." The statement "People don't want to buy a drill, they want to buy a hole" does not go deep enough into what the desired goal actually is. Why does a client want to buy a hole? She wants to install bookshelves. Why does she want to install bookshelves? She wants a place to conveniently store books. Why does she want to conveniently store books? Perhaps she wants to store information in an organized manner so she can more conveniently access that information and she is frustrated with her existing bookshelf. The creative solution may be something completely different than a bookshelf. Similarly, consumers do not want to buy HMDs. They might want to effectively train for specific tasks, or they may want to bring in more customers into their place of business. A VR demo might be to persuade potential customers to physically travel to the store to purchase merchandise that is largely unrelated to the VR demo.

In addition to continuously asking "why" questions, here are some examples of useful high-level questions to understand background and context.

- What is the inspiration/motivation? Is it meant to replace some existing system (a real-world system or an older VR system)? Or is it to create an entirely new experience?
- Who are the decision makers? Who are the principle stakeholders and intended beneficiaries?
- Where will the system be deployed? Will it be in a corporate office, at a theme park, or in people's homes?
- Is the VR project part of a larger vision? Is the project a brand new initiative, or is it being fit into an existing project? How is the VR project related to other projects? Even if not directly a part of a larger vision or project, the relationship between them may still be important.
- Does the project play a minor or major role in the organization? What impact will it have on the organization?
- What are the characteristics of the cultural, social, and politic aspects of the target organization or community? Military settings are very different from entertainment settings.
- Who will the project affect other than the users?
- How many users will be using the system? How many locations?
- What is the planned lifespan once deployed? Is the expectation for this to be a single project or will there be further work (assuming success) after completion?

Meta Platforms, Inc.
Exhibit 1067
Page 398

- What are the desired outcomes? How will success be measured?
- What are the resources and their limitations? What is the budget range and what is the timeframe of when the project must be completed?

More detailed questions should also be asked as one begins to better understand the context. Talking with subject-matter experts is essential, especially for non-entertainment applications. One qualified opinion from a subject-matter expert is often more valuable than 100 uninformed opinions (although a wider uninformed audience can be useful as well since they may be your future customers). Respect experts' time by preparing specific questions that are relevant to their expertise.

In addition to asking questions, it is helpful to see for yourself. Physically travel to the site to meet with insiders so they can explain what will likely be an alien world for you. Ask various individuals there whether they are employees or customers, what they are doing, why they are doing it, and how it might be done better. Carefully observe and take notes about peoples actions in addition to their words—their actions may indeed be more informative. If you can put yourself in their shoes, then you will understand. Ask questions not only to others but also ask questions to yourself. How do they currently perform tasks? If the goal is to better educate people about a topic then how do people currently learn about that topic? What barriers do you see that are keeping them from learning more effectively?

31.3 Assessment and Feasibility

As one learns more about the goal of the project, it is useful to step back and assess if VR is the right solution. Whereas VR has the potential to cross all disciplines and be useful in a wide variety of applications, that does not mean VR is the ideal solution to every problem. VR is certainly not the ideal tool for every problem today as the technology is still far from being perfected.

Example questions in assessing and therefore determining the feasibility of a VR project include the following.

- What results must be achieved for the project to be considered a success?
- What functionality must be supported? Can non VR technology support it better than today's VR technology?
- Will a VR solution fit within the real-world context such as the environment, organization, and culture?
- What are the competitive advantages to existing and/or non VR solutions? What are the alternatives and the trade-offs?

Meta Platforms, Inc.
Exhibit 1067
Page 399

- What is the expected profit, cost savings, and/or other added benefits?
- What are the hidden costs outside of project development? Examples include shipping or deployment costs, user-training costs, marketing and sales, and maintenance.
- Is the project possible given the budget, time, and other resources?

The answers to these questions are often not binary. Often, a mix of the real world and VR is optimal (e.g., traditional education mixed with practice in a simulator).

31.4 High-Level Design Considerations

The user experience is the end of a chain of design considerations and the context in which different design options are considered is important. It is useful to think about design from three different points of view (Bolas, 1989, 1992).

Design *with* virtual environments focuses on the use of VR to help solve an existing problem or to create a new invention. Carefully considering application needs can drive the design when VR is used as a design tool itself, for example scientific data visualization or automotive design. If done well, VR can amplify human intelligence during the investigation and solving of real-world problems (e.g., enabling easier matching and recognition of complex patterns). With this approach, VR systems should complement, not replace, other tools and forms of media in order to maximize insight.

Design *for* virtual environments focuses on improving the hardware and software of VR systems themselves. With this approach, the designer should carefully consider the technology affordances used to provide the experience. For example, considering input device characteristics and classes (Chapter 27) can help the designer choose, modify, or upgrade specific hardware. Considering different interaction patterns and techniques (Chapter 28) can help to improve existing implementations and to develop new interaction metaphors.

Design *of* virtual Environments focuses on the creation of completely synthetic environments—the virtual worlds. This approach works well for artistic mediums and entertainment applications. Part IV focuses on creation of content.

31.5 Objectives

Objectives are high-level formalized goals and expected outcomes that focus on benefits and business outcomes over features. Focusing on benefits/outcomes enable the team to work toward the vision and solve problems instead of implementing features

Meta Platforms, Inc.
Exhibit 1067
Page 400

that users may or may not care about. Features are usually more preferred by engineers as they are easier to implement but may not be what users care about. Yet, engineers also do not like to be micromanaged. Thus it is best to let the engineers do what they do best—solve problems through the use of features they came up with that help to achieve the objectives. The team will gain insight into the value of features as they are being built and tested. If a feature ends up not moving the project toward the objectives, it can be changed, removed, or replaced.

Benefits that objectives describe often deal with time and/or cost savings, revenue generation, lower safety risks, increase in user productivity, etc. Note that objectives are different than requirements described in Section 31.15. Requirements are more specific and often more technical (e.g., latency and tracking requirements).

Good objectives include what is described by the acryonym **SMART**—Specific, Measureable, Attainable, Relevant, and Time-bound:

Specific. The objective should be clear and unambiguous; state exactly what is expected. The objective might also include specifically why it is important.

Measureable. The objective should be concrete so that progress toward the goal and ultimately success can be determined in an objective manner. This is essential for the Learn Stage (Chapter 33) where measurement and testing occurs.

Achievable. The objective should somehow be feasible. The description of how is not important here, just the belief that it can be achieved in some manner.

Relevant. The objective should matter. This does not necessarily directly affect the end result; the objective might support other objectives that do affect the end result.

Time-Bound. An objective should state a date of when it will be completed.

An example of an objective is "The VR training system will be deployed on January 1, 2016, and will result in a 30% increase in productivity as defined in Section X after three sessions of training." The 30% productivity goal is a number that can be worked toward before delivery of the system through user testing.

A long list of objectives may be quite aggressive. Rarely can multiple aggressive goals be achieved in the first delivery nor should they. Prioritize objectives for the first delivery and save other objectives for later delivery. It is better to deliver 50% of the highest priority objectives than to achieve 50% success on each individual objective.

31.6 Key Players

An important early step is to identify and enroll key players. A **key player** is someone that is essential to the success of the project. Key players might be stakeholders, part-

Meta Platforms, Inc.
Exhibit 1067
Page 401

ners, the client or sponsor, government agencies, customers, subject-matter experts, end users, marketing experts, usability experts, consultants, business developers, engineers, etc. It is vital the core key players are in agreement with the vision, really care about the project, and are committed to achieving success. Identifying and enrolling the *right* key players can be a huge undertaking in itself. This might include pitching to investors, writing business or proposals, assessing client needs, asking for referrals, recruiting others to join the team, etc.

Embracing key players often take one of two paths.

Enroll others in the vision. Find key players that are inspired by the vision and believe it can be accomplished, and move on when the individuals are not right for the project. Don't waste time on those that are not a good fit. This path is most common when creating an entertainment experience or when recruiting others to join the team. Note that just because an individual or small group originally creates the vision does not mean they should not seek input from enrolled key players.

Help to solve a need. Identify pain points and work toward finding a way to solve problems and satiate existing desires. This path is common in industry, for example to help reduce costs or find better ways to train employees. Demonstrating how you can help individuals (especially the decision makers) with their specific problems makes all the difference in converting skeptics to key players.

In both cases, be prepared to do a lot of pitching and proposing. Realize in advance that many people may not have an interest no matter how good of a fit you think they might be. Listen to what they have to say (even if they do not want to be involved they may have valuable input) and then move on. Do not waste time chasing such individuals.

31.7 Time and Costs

Accurate estimates are extremely difficult, even for those with extensive project experience. Early estimates should not be expected to be accurate. Because of this, it often makes sense to separate out initial assessment and feasibility from implementation when negotiating contracts. Contracts might also be written with milestones, with the client making additional payment as milestones are achieved and with the expectation that the contract might be renegotiated at each milestone as more is learned and estimates are refined. If a project is late, it is not a good idea to add additional team members as Brooks' Law states "adding manpower to a late software project makes it later" (Brooks, 1995). Because time, budget, and quality are often non-negotiable, the

Meta Platforms, Inc.
Exhibit 1067
Page 402

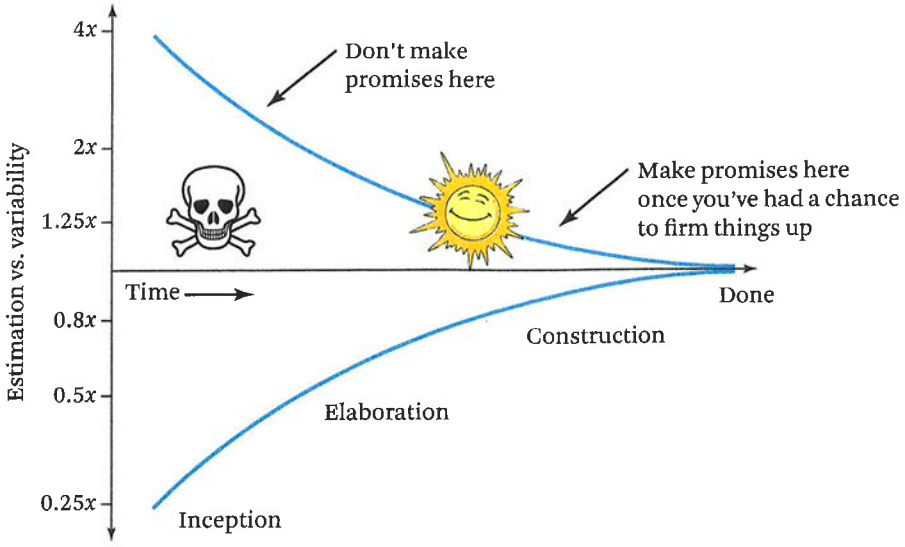


Figure 31.1 **Estimating development is often incorrect at the beginning. It is not uncommon for estimates to be off by a factor of four.** (Based on Rasmusson (2010))

scope of the project may need to be scaled back when a project is taking longer than expected.

Poker planning is a game of estimating development effort (Cohn, 2005). Although it is best done when planning for implementation, it can also be done at a higher level early in the project with the understanding the estimates will not be as refined since the project is not yet well defined. The team sits at a table, a specific task is stated, and all players write down an estimate of the time they think it will take to complete the task. After secretly writing the estimated time on the card, players face the card down on the table and all cards are flipped right side up. If all estimates are in the same general range, then the average is taken. Planning poker is not a voting system—three short estimates do not out vote one longer estimate. If there are outliers then those outliers are discussed. It may be the case that the person who wrote the outlier considered something other team members had not.

Initial development estimates are often off by as much as 400% (Rasmusson, 2010) and developers tend to overestimate their abilities even when they know they tend to overestimate. Estimates improve as implementation continues and more is learned about the project (Figure 31.1). One way to determine a better estimate is to track the ratio of how long tasks are taking relative to the initial estimate. Future task estimations can be improved by multiplying the tracked ratio by the initial task estimates.

Meta Platforms, Inc.
Exhibit 1067
Page 403

For example, if tasks estimated to take 5 days actually take 15 days then the ratio is 3. Future tasks estimated to take 5 days will also likely take 15 days and tasks estimated to take 10 days will likely take 30 days.

31.8 Risks

The intention of identifying risk is to bring awareness of any danger that might affect the project so that appropriate action can be taken to mitigate that risk. After brainstorming with the team on all possible risks, divide listed risks into two groups: (1) risk the team has influence or control over and (2) risk the team has no influence or control over. Understanding risk that is outside the teams control is important for determining if the project is worth pursuing. Once such risk is understood, the leadership must decide to fully commit (or end the project). Then the team can focus on minimizing controllable risk.

Risk is especially important to be aware of for VR projects because there are so many unknowns and technology is improving so quickly. Risk increases exponentially as project size and duration increases (Figure 31.2). A two-year project may quickly be put out of business by a new startup that didn't even exist at the beginning of the project. To minimize risk, projects should be kept as short as possible. This does not mean larger projects should not be undertaken. It does mean to break larger projects

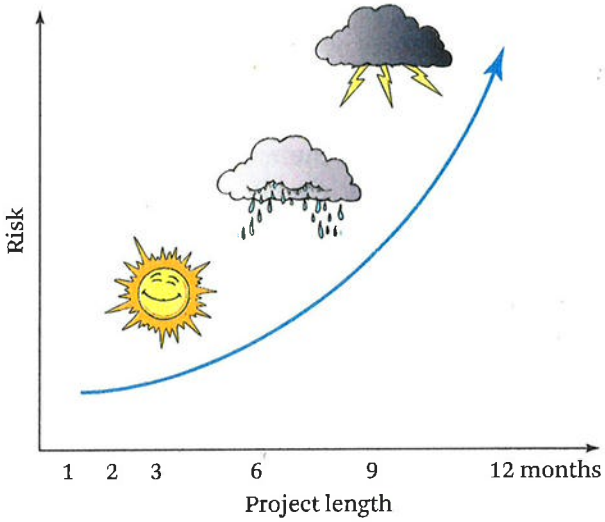


Figure 31.2 **The risk of a VR project failure increases exponentially over time.** (Based on Rasmusson (2010))

Meta Platforms, Inc.
Exhibit 1067
Page 404

down into smaller subprojects. After successful completion of the smaller project, then new contracts and extensions can be made.

31.9 Assumptions

Assumptions are high-level declarations of what one or more members of the team believe to be true (Gothelf and Seiden, 2013). Everything created by humans begins with assumptions, whether the creators consciously realize those assumptions exist or not. VR projects are no different.

Explicitly looking for and declaring assumptions allow team members to begin a project from a common starting point. Get the team together and carefully go through the project description and dissect its assumptions. The team will likely find that what they assume are shared assumptions, are actually different assumptions. Be bold and precise in listing assumptions, even when not sure. Wrong explicit assumptions are much better than vague assumptions. Wrong assumptions can be tested; vague assumptions cannot (Brooks, 2010).

Examples of assumptions are who the targeted users are, user needs/desires, what the greatest challenges are, what are the important VR elements of the experience, etc. The list of assumptions should be quite long. Unfortunately it is not feasible to test all assumptions, so prioritize the assumptions with the riskiest assumptions listed first.

31.10 Project Constraints

All real-world and virtual-world projects have constraints that must be designed around. Project constraints can come from a wide range of sources. For example, a constraint might be limited hardware functionality (e.g., tracking may only be reliable within some volume), a maximum amount of accepted latency (e.g., 30 ms is low enough to do prediction fairly well), or budget/time constraints (limiting the complexity of implementation). A single individual should be responsible for tracking/controlling constraints, with transparency to the entire team.

Project constraints at first seem harmful to what can be done. Whereas it is true that for building a mediocre experience, it is easier to do a general-purpose design with few constraints (e.g., put the user in a world and have him fly around) than to do a special-purpose design (Brooks, 2010). However, for a high-quality experience, it is easier to do a special-purpose design because establishment of the problem has already started—it is more a task of discovering constraints. Discovering existing constraints provides clarity and focus resulting in higher quality experiences that can be more effectively built. Constraints also provides a basis for feedback (i.e., constraints can help focus questions) and challenge the team, stimulating fresh creations.

Meta Platforms, Inc.
Exhibit 1067
Page 405

31.10.1 Types of Project Constraints

Explicity listing project constraints can be very helpful in narrowing the design space. When listing constraints, consider the following (Brooks, 2010).

Real constraints are true impediments that cannot be changed. Examples include physical barriers, rules outside of the team's control, the number of controllers on a button, and the tracked volume of a space.

Resource constraints are real-world limitations of supply. For any project, there is always at least one scarce resource that must be rationed or budgeted. Examples are dollars, deadlines, minimum hardware specs users are expected to own, end-to-end system latency, and battery life for mobile VR. After listing all possible resource constraints, prioritize them, as there will always be more to do. Some of the low priority constraints might be moved to misperceived constraints (described below). Track resource constraints with transparency to the entire team, and have a single individual control firmly.

Obsolete constraints were once real constraints but are no longer valid. This can be due to a change in rules or due to the ever-ending improvement of technology. Examples include better tracking, faster CPUs, more buttons on a controller, and number of reliably detected gestures.

Misperceived constraints are perceived to be real but are not. They are often are so embedded in our lives that we don't even realize that we treat them as constraints. VR is full of misperceived constraints because many of the rules of the real world do not apply to VR. The designer can choose to allow users to walk through walls or reach a greater distance than what can be done in the real world.

Indirect constraints are the side effects of real constraints. These constraints are not necessarily real constraints as these indirect constraints are based on assumptions of how something is achieved. Indirect constraints are important to differentiate because if they are seen as real constraints, then solutions are cut off. The number of polygons for a scene is an indirect constraint. The true constraint is the frame rate as rendering polygons can be optimized.

Intentional artificial constraints are added by the designer to narrow the design and to enhance the user experience. Adding constraints to a VR experience can significantly improve interaction. Section 25.2.3 discusses the power of adding interaction constraints.

Meta Platforms, Inc.
Exhibit 1067
Page 406

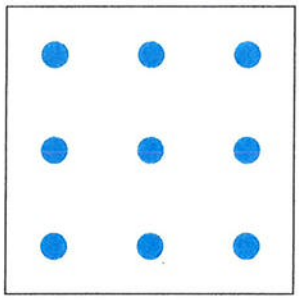


Figure 31.3 **A design puzzle. Draw over all nine dots with four straight lines connected end-to-end. Figure 31.5 shows a solution.**

31.10.2 A Constraints Puzzle

Take a look at Figure 31.3 and try to draw over all nine dots with no more than four straight lines connected end-to-end.

If you can't find a solution, then go through an exercise of explicitly listing the constraints in the bullet points listed above. You may find a solution once you explicitly list all of the different types of constraints. If not, go back and think more carefully about any misperceived constraints. If you are able to do this (and you haven't seen the solution before) then congratulations! There are very few people who are able to solve this without first seeing one of the solutions. Did explicitly listing out the constraints (specifically misperceived constraints) help? A reason why most people can't solve this puzzle is because most people don't take the time to explicitly think about constraints.

Now try to draw over all nine dots with three straight lines connected end-to-end (hint: think even further outside the box). Figure 31.7 shows a solution.

Now try to draw over all nine dots with three straight lines but this time the lines cannot extend beyond the box boundaries. If you think that is impossible then go back to your list of constraints and update that which has changed. Figure 31.8 shows a solution.

Now try to figure out how you could draw over all nine dots with a single straight line. Note you may be constrained by the tool you are using to solve the problem, instead of a constraint inherent to the problem. List any constraints on tools that people normally use to solve such puzzles and then try again. Figure 34.1 shows a solution.

Meta Platforms, Inc.
Exhibit 1067
Page 407



Name	• Job • Experience • Activities • Attitude • Competencies • Age
• Problems • Pain points • Needs • Concern • Fears • Desires	• Knowledge of VR • Dream VR system • Vision of VR • VR hardware access • Budget for VR • Activities that fit VR

Figure 31.4 **Persona template.**

31.11 Personas

Attempting to design for the entire population is difficult as users vary widely in their ability to use VR systems, making generalizations difficult (Wingrave et al., 2005; Wingrave and LaViola, 2010). As a result, targeting more specific users makes the design easier. Personas help do this.

Personas are models of the people for who will be using the VR application. Getting clear on users by explicitly defining personas helps to prevent the design from being driven by design/engineering convenience where users are expected to adapt to the system. Not only should the application be defined and designed around these personas, but representative users can be targeted to collect feedback for the Learn Stage (Chapter 33).

Divide a notecard into four quadrants. Start by getting the team together and make the persona up by providing a simple sketch and name in the upper left quadrant. Add a basic description of the person in the upper right quadrant. Add different types of challenges the person has in the lower left quadrant. Then add information about how the person relates to VR in the lower right corner. Figure 31.4 shows an example template that might be used. Do this for 3–4 characters that represent the full range of your targeted users.

During later iteration cycles, personas should be validated and modified as more is learned about real users. If personas are especially important (e.g., for therapy

Meta Platforms, Inc.
Exhibit 1067
Page 408

applications) then data should be collected with interviews (Section 33.3.3) and/or questionnaires (Section 33.3.4).

31.12 User Stories

User stories emerged from agile development methods and are short concepts or descriptions of features customers would like to see (Rasmusson, 2010). They are often written on small index cards to remind the designer to not go into too much detail. We don't know yet if we're actually going to need or implement that feature. They are written from the user's point of view and should be written with the client and multiple team members.

User stories are written in the form "As a <type of user> I want <some goal> so that <some reason>." This defines who, what, and why. User stories often turn into requirements and constraints. Break big story items into smaller manageable stories. Go through the list and clean it up, remove duplicates, group like items together, and turn it into an action item or todo list.

Ideally, user stories fit with the Acronym **INVEST** created by Bill Wake (Wake, 2003): Independent, Negotiable, Valuable, Estimable, Small, and Testable.

Independent user stories are easier to develop for when they do not overlap with other user stories. This means the stories can be implemented in any order and the creation or modification of one story does not affect another story. This can be a challenge for VR because interactions often affect other interactions.

Negotiable user stories are high-level descriptions that are modifiable. VR components must be modifiable because what we think will work well does not always work well, and what works can only be determined by learning from users who actually try the implementations.

Valuable user stores are specifically focused on giving value to the user and in a language that is understandable by anyone. A brilliantly implemented framework has little value if it does not offer value to users.

Estimable user stories are understandable so that implementation can be estimated. If the story is too complex then it should be broken down into easier-to-understand estimable stories.

Small user stories enable fast implementation. If a story is too big to be quickly implemented it should be broken down into smaller stories.

Meta Platforms, Inc.
Exhibit 1067
Page 409

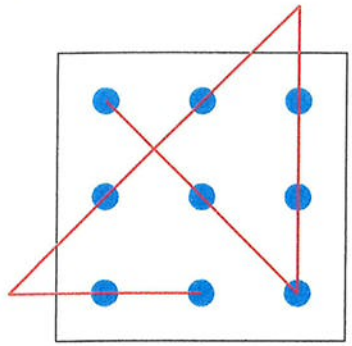


Figure 31.5 **A solution to Figure 31.3. By realizing the surrounding box is a misperceived constraint, one can think outside of the box. Now try doing the same with three straight lines (solution shown in Figure 31.7).**

Testable user stories are written so that a simple experiment can clearly determine if the implemented story is working or not.

31.13 Storyboards

Storyboards are early visual forms of an experience. They are especially good at getting the point across to those only loosely involved with the project. Figure 31.6 shows an example of a storyboard.

Traditional software storyboards typically end up being screenshots more than interactions which are closer to a mockup than an experience. Storyboards are more useful for VR because the user can be shown directly interacting with objects—experiences and interactions can be quickly sketched and conveyed without having to worry about screen layout details.

Storyboarding can however be more difficult for VR than for video storyboards because of the non-linear nature of VR. In many other cases there might be multiple connections between storyboard frames.

31.14 Scope

When defining the project, explicitly stating what will not be done can be just as helpful in stating what will be done (Rasmusson, 2010). This will set clear expectations for all involved and allow the team to focus on what is important. Items defined to be in scope are what are nonnegotiable and must be completed. The team should explicitly

Meta Platforms, Inc.
Exhibit 1067
Page 410

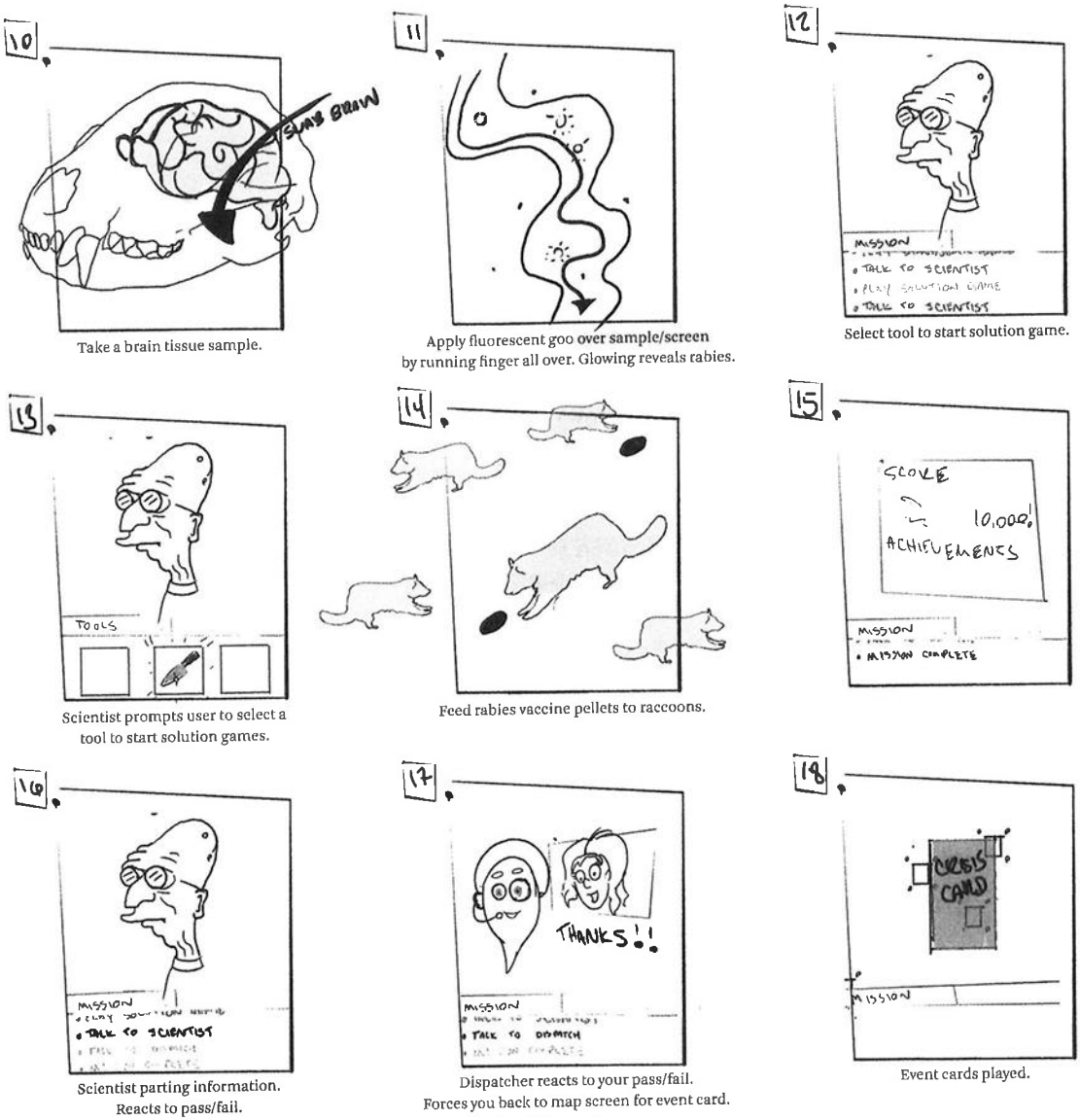


Figure 31.6 Part of an example storyboard for an educational game. (Image courtesy of Geomedia)

Meta Platforms, Inc.
Exhibit 1067
Page 411

Table 31.1 **Defining what is in and out of scope makes it clear to all parties involved what will and will not be done.**

In Scope	Out of Scope
Bimanual tracking	Tracking for standing and walking
Ray selection	Networked environment
Embodied avatar	Voice recognition
Hand-held panel	Fine-tuning of colors
Unresolved	
Travel technique	
What widgets/tools to include on the panel	

not be concerned with items that are out of scope; they might or might not be included in later releases.

31.15 Requirements

Requirements are statements that convey the expectations of the client and/or other key players such as descriptions of features, capabilities, and quality. Each requirement is a single thing the application should do. Requirements come from other parts of the Define Stage such as assumptions, constraints, and user stories. Like other parts of the Define Stage, the client and/or other key players should be actively involved in defining requirements. Requirements do not specify implementation.

Requirements aid with the following.

- Requirements communicate mutual understanding between the client and contractor, as well as any other parties involved. Requirements should be written in easy-to-understand language that can be understood by everyone involved with the project.
- Requirements help to organize, understand, and decompose goals of the project into more explicit descriptions.
- Requirements act as input to the design specification (Section 32.2).
- Requirements are used as a source for determining if a system is working as expected (Chapter 33).

A requirements document should be as minimal as possible without compromising clarity. Long requirements documentation end up rarely being read entirely by anyone other than the person writing the document, and that might not even occur due to

Meta Platforms, Inc.
Exhibit 1067
Page 412

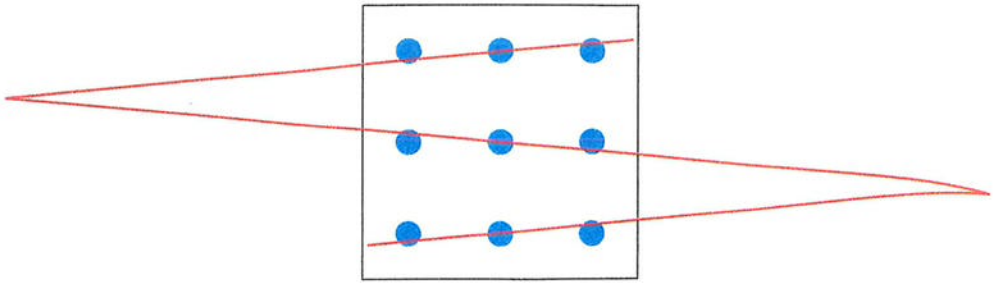


Figure 31.7 **A solution to Figure 31.3 using only three lines. Now try draw to draw over all nine dots with three straight lines but this time the lines cannot extend beyond the box boundaries (solution shown in Figure 31.8).**

different groups writing different sections. Not reading the document in its entirety can be a major problem due to key players not being clear on understanding the entire project.

Each individual requirement should be complete, verifiable, and concise, yet with room for innovation and change. Requirements may be changed and clarified throughout the project as initially some requirements may be ill-defined and others might become discovered as the project evolves. Not proceeding until all requirements are written down would mean never starting. The Agile manifesto states "Any attempt to formulate all possible requirements at the start of a project will fail and would cause considerable delays" (Pahl et al., 2007). Work with the client and/or other key players when changes are necessary.

31.15.1 Quality Requirements

Quality Requirements (also known as non-functional requirements) define the overall qualities or attributes of a system or application. Quality requirements can be thought of as a definition for quality of service or quality control. Quality requirements may place restrictions or constraints on the solution such as usability, aesthetics, security, reliability, and maintainability. Common quality requirements include system requirements, task-performance requirements, and usability requirements as described below.

System requirements describe parts of the system that are independent of the user, such as accuracy, precision, reliability, latency, and rendering time.

Accuracy is the quality or state of being correct (i.e., closeness to the truth). A system with consistent systematic error has bias and low accuracy. For example,

Meta Platforms, Inc.
Exhibit 1067
Page 413

distortion in a tracker system has low accuracy because if a user moves in one direction, the tracked point might unexpectedly and wrongly move the tracked point in a different direction.

Precision is the reproducibility and repeatability of getting the same result. A tracked tool that shakes or jitters in the hand due to bad tracking has low precision and results in difficulty selecting small objects or menu items.

Reliability is the consistency that some part of the system works. Often measured as a hit rate or failure rate. For example, a tracking system might be required to lose tracking no more than once per 10 minutes.

Latency is the time a system takes to respond to a user's action (Chapter 15).

Rendering time is the amount of time it takes for the system to render a single frame. A requirement might be that rendering time not exceed 15 ms for any single frame to ensure frames are not repeated.

Task performance requirements focuses on how effectively a task can be completed. Task performance is the objective quality of how the task is performed by users, such as time to completion, performance accuracy, performance precision, and training transfer.

Time to completion is how fast a task can be completed. This is typically measured as the average time it takes for users to complete the task.

Performance accuracy is the resulting correctness of a user's intention. A manipulation or navigation task has decreased accuracy as the distance from the desired position or path increases.

Performance precision is the consistency of control that users are able to maintain. It can be thought of as the fine-grained control afforded by the technique (McMahan et al., 2014). Travel is precise if it enables travel along narrow paths.

Training transfer is how effectively knowledge and skills obtained through the application transfer to the real world. Training transfer is influenced by many factors such as how well the stimuli and interactions match the real world (Bliss et al., 2014).

Usability requirements describe the quality of the application in terms of convenient and practicable use. Common usability requirements include ease of learning, ease of use, and comfort (McMahan et al., 2014).

Meta Platforms, Inc.
Exhibit 1067
Page 414

Ease of learning is the ease with which a novice user can comprehend and begin to use an application or interaction technique. Ease of learning is often measured by the time it takes a novice to reach some level of performance, or by characterizing performance gains as use increases.

Ease of use is the simplicity of an application or technique from the user's perspective and relates to the amount of mental workload induced upon the user of the technique. Ease of use is typically measured through subjective self-reports, but measures of mental workload are also used.

User comfort is a state of physical ease and freedom from sickness, fatigue, and pain. Different input devices and interaction techniques can affect user comfort and making small changes can have a big impact. User comfort is especially important for experiences that are longer than a few minutes. User comfort is typically self-reported.

31.15.2 Functional Requirements

Functional requirements specify what some part of the system must enable the user to achieve and often include a set of inputs, behaviors, and outputs. Functional requirements are typically more dependent on the specifics of the project than quality requirements. Detailed functional requirements often evolve out of task analysis (Section 32.1) and use cases (Section 32.2.3). Examples of functional requirement include the following.

- Computer-controlled characters will navigate to the designated position via the shortest path, where all portions the path will have (1) a slope between −20° and 20°, and (2) a width greater than 0.8 m.
- All objects marked as selectable will be able to be directly selected by the user via intersecting the hand geometry with the object and pushing the grab button.
- Users will not be able to navigate closer to geometry than 0.2 m. in the horizontal when that geometry is at a height between 0.2 meters and 2.4 m. above the travel plane.

31.15.3 Universal VR Requirements

Listed below are some requirements that should be considered for all fully immersive VR applications. These requirements should be achieved and maintained from the start of development as optimization can be a challenge late in the project (e.g., reducing scene complexity can require new art assets). The moment these requirements are not being met, the software should automatically and immediately recognize the

Meta Platforms, Inc.
Exhibit 1067
Page 415

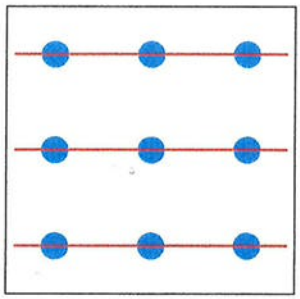


Figure 31.8 **A solution to Figure 31.3 but with an obsolete constraint removed. The new problem statement did not say the lines had to be connected. Now try to draw over all nine dots with a single line (solution shown in Figure 34.1).**

problem and attempt to resolve the problem by reducing the scene and/or rendering complexity (e.g., transition to simpler lighting algorithms). The less complex settings should then be maintained instead of alternating back and forth as variable latency can cause sickness and prevent adaptation (Section 18.1). If the requirements still cannot be met than the scene should fade out and/or transition to a simpler scene or video-see through if available (assuming video-see through can achieve the requirements). The fading out of the scene during development will be frustrating to developers but will ensure quality as they will be motivated to fix the problem.

End-to-end system delay will not fall below 30 ms. The scene will automatically fade out if this requirement cannot be maintained.

Minimum frame rate will be the refresh rate of the HMD (60–120 Hz depending on the HMD). The scene will automatically fade out if this requirement cannot be maintained.

The screen will fade out if and when head tracking is lost. The scene will automatically fade out if this requirement cannot be maintained.

Any camera/viewpoint motion not controlled by the user will not contain accelerations for periods longer than one second. Such motions should only rarely occur, if ever.

Input devices will maintain 99.9% or better reliability. Anything less is frustrating to users.

Meta Platforms, Inc.
Exhibit 1067
Page 416

The Make Stage

The **Make Stage** is where the specific design and implementation occurs. This stage is arguably the most important part of creating VR experiences. The other stages would only be dreams and theory without the Make Stage whereas at least something, albeit perhaps not good, can be used and experienced with only the Make Stage. The Make Stage is also often where most of the work occurs. Fortunately the implementation of VR experiences is in many ways similar to creating other software once the experience is properly defined and feedback is obtained. The same logic, algorithms, engineering practices, and design patterns are still used. One primary difference is that there is more emphasis on working with hardware. Makers should not develop VR applications without full access to the VR hardware the system is designed for because the VR experience is so tightly integrated with that specific hardware.

The Make Stage often consists of primarily using existing tools, hardware, and code where the focus is on gluing the different pieces together so that they work together in harmony. This is the case for most VR projects, as development tools such as Unity and the Unreal Engine have proven to be very effective for building a wide variety of VR worlds. In some cases, the project may require implementing from the ground up. Examples of when that might be the better option is when a true real-time system is required, in-house code with specialized functionality already exists, or specialized/optimized algorithms are required (e.g., volumetric rendering). However, the default should be to use existing frameworks and tools unless there is a good reason not to do so. Resistance to change is no excuse to not using more effective technology. Developers will find the learning curve of using modern tools quite small relative to the benefits.

This chapter discusses task analysis, design specification, system considerations, simulation, networked environments, prototyping, final production, and delivery.

Meta Platforms, Inc.
Exhibit 1067
Page 417

32.1 Task Analysis

Task analysis is analysis of how tasks are accomplished or will be accomplished, including both physical actions and cognitive processes. Task analysis is accomplished by insight gained through understanding the user, learning how the user currently performs tasks, and determining what the user wants to do. Task analysis provides organization and structure for description of user activities, which make it easier to describe how the activities fit together.

Task analysis processes have become complex, fragmented, difficult to perform, and difficult to understand/use (Crystal and Ellington, 2004). The goal of task analysis should be to organize a set of tasks so they can be *easily and clearly* communicated and thought about in a systematic manner. If the result of the task analysis complicates understanding through extensive documentation then the analysis has failed. Task analysis should be kept simple without requiring learning specialized processes or symbols. Part of the reason to do a task analysis is to communicate how interaction works, and anyone should be able to understand the result without having to have specialized training in reading a specific form of task analysis diagrams.

32.1.1 When to Do Task Analysis

The first iteration of task analysis is done when trying to understand how a task is or will be performed. When the goal for the VR application is to replicate actions in the real world (e.g., training applications), task analysis should start with the actual tasks performed in the real world in order to understand what tasks must be carried out, to document proper descriptions of activities, and to look for how to improve the process. Applications built without doing a full task analysis may have low validity relative to the real world due to the task being simplified in the virtual environment (Bliss et al., 2014). In other cases, task analysis might be used in creating a new magic VR interaction technique that is very different than any real-world task.

Task analysis helps to define the design and to realize early the implications of redesign. Task analysis results in more than just straightforward understanding of simple tasks; it helps to create understanding of the relationship between different tasks, how information flows, and how users affect task sequence by their decisions. This understanding can help to prioritize tasks and to determine what can be automated without user intervention. Without task analysis, designers might be forced to guess or interpret desired functionality that most often leads to poor design. Evaluation plans and validation criteria also depend on task analysis.

Task performance can be used for many purposes. One use is to understand, modify, and create new interaction techniques. This can be done by decomposing the

Meta Platforms, Inc.
Exhibit 1067
Page 418

techniques into subtask components, which enables comparison of subtask components rather than comparing holistic techniques. One can then replace subtask components with other (rather already used elsewhere or new methods) components from other techniques.

Task analysis is also used at later stages of development to determine how the current implementation deviates from the design/intended solution, and the consequences. Like all aspects of VR development, task analysis should be flexible and iterative, allowing for modification during all but the final stages of production.

32.1.2 How to Do Task Analysis

There are many ways to do task analysis. Here steps are generalized into steps of finding representative users, task elicitation, organize and structure, review with the representative users, and iteration.

Find Represenative Users

Although in practice task analysis is initially done by the team, it is important to observe and interview representative users. Those matching the personas discussed in Section 31.11 should be sought out to make sure the analysis represents activities of the user population. Use more than one person to ensure the analysis represents activities represented by the entire user population. This can be a time-consuming process and outsourcing someone to find such individuals allows the team to focus on what they are good at.

Task Elicitation

Task elicitation is gathering information in the form of interviews, questionnaires, observation, and documentation (Gabbard, 2014).

Interviewing (Section 33.3.3) is verbally talking with users, domain experts, and visionary representatives—this provides insight into what users need and expect. Questionnaires (Section 33.4) are generally used to help evaluate interfaces that are already in use or have some operational component. Observation is watching an expert perform a real-world task or a VR user trying a prototype (this resembles formative usability evaluation—see Section 33.3.6). Documentation review identifies task characteristics as derived from technical specifications, existing components, or previous legacy systems.

To collect information efficiently, one should plan/structure the process. Focus on activities that are currently most relevant and start with a description of an activity that begins with a rough description of what to do. Carefully thinking about the different stages of the Cycle of Interaction (Section 25.4) is a great place to start. Ask "how"

Meta Platforms, Inc.
Exhibit 1067
Page 419

questions to break the task into subtasks for more detail (but don't get bogged down with details—know when enough is enough). Ask "why" questions to get higher-level task descriptions and context. "Why" questions and asking what happens before and what happens after can also help to obtain sequential information. Use graphical depictions with those being talked with to collect higher quality information.

Organize and Structure

The most common form of task analysis is **hierarchical task analysis** where tasks are decomposed into smaller subtasks until a sufficient level of detail is reached. Each node below a task in the hierarchy addresses a single subtask. Each subtask can be thought of as a question that must be answered by the designer, and the set of subtasks is the set of possible answers for that question. This is a good place to start as it works well for understanding the details of how users perform actions. High-level task descriptions go on the top with those tasks broken down into lower-level task descriptions. Sequence of tasks is ordered from left to right. Don't be overly concerned about getting this perfect or thinking there is a right way to do it—like everything in iterative design, you will iterate toward something better. The specifics may depend on the project and you can use your own style such as user activities being represented by boxes and system activities represented by circles, and relationships between boxes/circles represented by arrows.

Hierarchical task analysis has its limitations as the hierarchical pattern limits what can be described with tasks. In addition to hierarchical diagrams, tables, flow charts, hierarchical decomposition, state transition diagrams, and notes can be used where appropriate. Task analysis might also include defining affordances, and the corresponding signifiers, constraints, feedback, and mappings (Section 25.2). For activities that are performed multiple times, generalize the activity to a pattern and give it a name. Then that type of activity can be represented by single node in different locations. If the task deviates from the general pattern give it a secondary name and/or a note of specific differences.

Review with Users

Once data has been organized and structured, one should review with the users who information was elicited from to verify understanding.

Iteration

Task analysis provides the basis for design in terms of what users need to be able to do. This analysis feeds into other steps of the Make Stage, and these Make Stage steps along with lessons acquired from the Learn Stage will feed back into refining the

Meta Platforms, Inc.
Exhibit 1067
Page 420

task analysis. Like everything else in iterative design, changes to the task analysis will occur. Document the latest changes so it is obvious what changed. Different colors convey such change well.

32.2 Design Specification

Design specification describes the details of how an application currently is or will be put together and how it works. Design specification is the between stage of defining the project and implementation, and is often closely intertwined or simultaneous with prototyping. In fact, in many cases the person doing the design is also doing much of the prototyping.

Specifying the design isn't just conducted to satisfy that which was created in the Define Stage, but elicits previously unknown assumptions, constraints, requirements, and details of task analysis. **Early wanderings** are the exploration of radically different designs early in the process before converging toward and committing to the final solution (Brooks, 2010). As new things are discovered, the corresponding defining documentation should be updated as appropriate.

Some of the most common tools for building VR applications are described in this section: sketches, block diagrams, use cases, classes, and software design patterns.

32.2.1 Sketches

A **sketch** is a rapid freehand drawing that is not intended as a finished work but is a preliminary exploration of ideas. Sketching is an art and very different from computer-generated renderings. Bill Buxton discusses what characteristics make good sketches in his book "Sketching User Experiences" (Buxton, 2007):

Quick and Timely. Too much effort should not be put into creating a sketch and it should be able to be quickly created on demand.

Inexpensive and disposable. Sketches should be able to be made without concern of cost or care if it is selected for refinement.

Plentiful. Sketches should not exist in isolation. Many sketches should be made of the same or similar concepts so that different ideas can be explored.

Distinct gestural style. The style should convey a sense of openness and freedom so that those viewing it don't feel like it is in a final form that would be difficult to change. An example is the way edges typically do not perfectly line up that distinguishes a sketch from a computer rendering that is tight and precise.

Meta Platforms, Inc.
Exhibit 1067
Page 421

Minimal detail. Detail should be kept to a minimum so that a viewer quickly gets the concept that is trying to be conveyed. The sketch should not imply answers to questions that are not being asked. Going beyond good enough is a negative for sketching, not a positive.

Appropriate degree of refinement. Detail should match the level of certainty in the designer's mind.

Suggestive and explorative. Good sketches do not inform but rather suggest, and results in discussion more than presentation.

Ambiguity. Sketches should not specify everything and should be able to be interpreted differently with viewers seeing relationships in new ways, even by the one doing the sketching.

Figure 32.1 shows a sketch drawn by Andrew Robinson of CCP games with all of these characteristics from the early phases of the VR Game EVE Valkyrie.

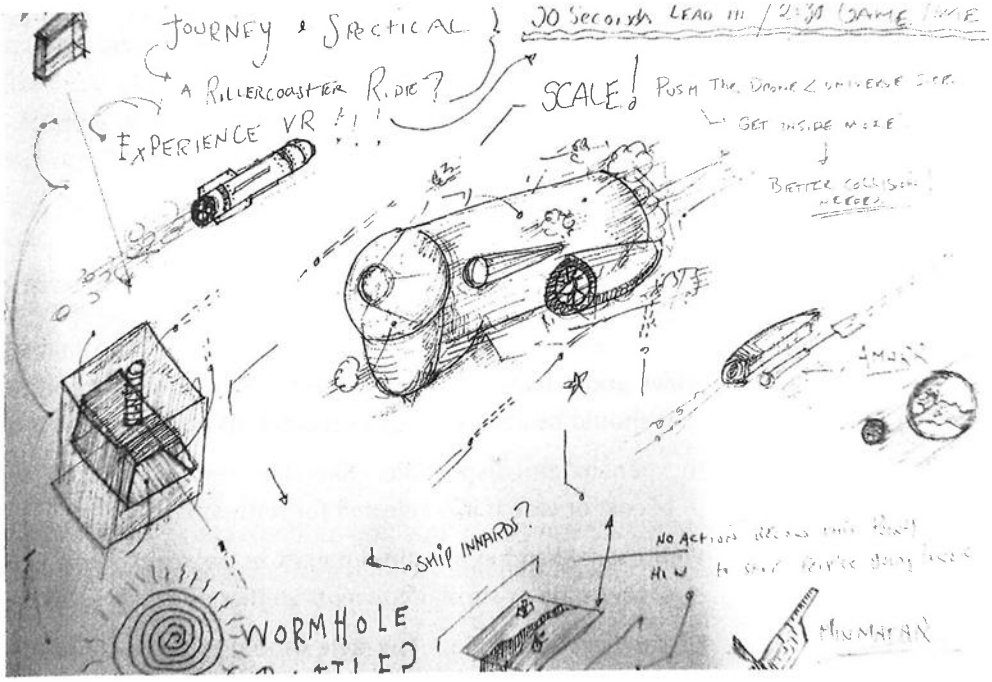


Figure 32.1 **Example of an early sketch from the VR Game EVE Valkyrie.** (Image courtesy of Andrew Robinson of CCP Games)

Meta Platforms, Inc.
Exhibit 1067
Page 422

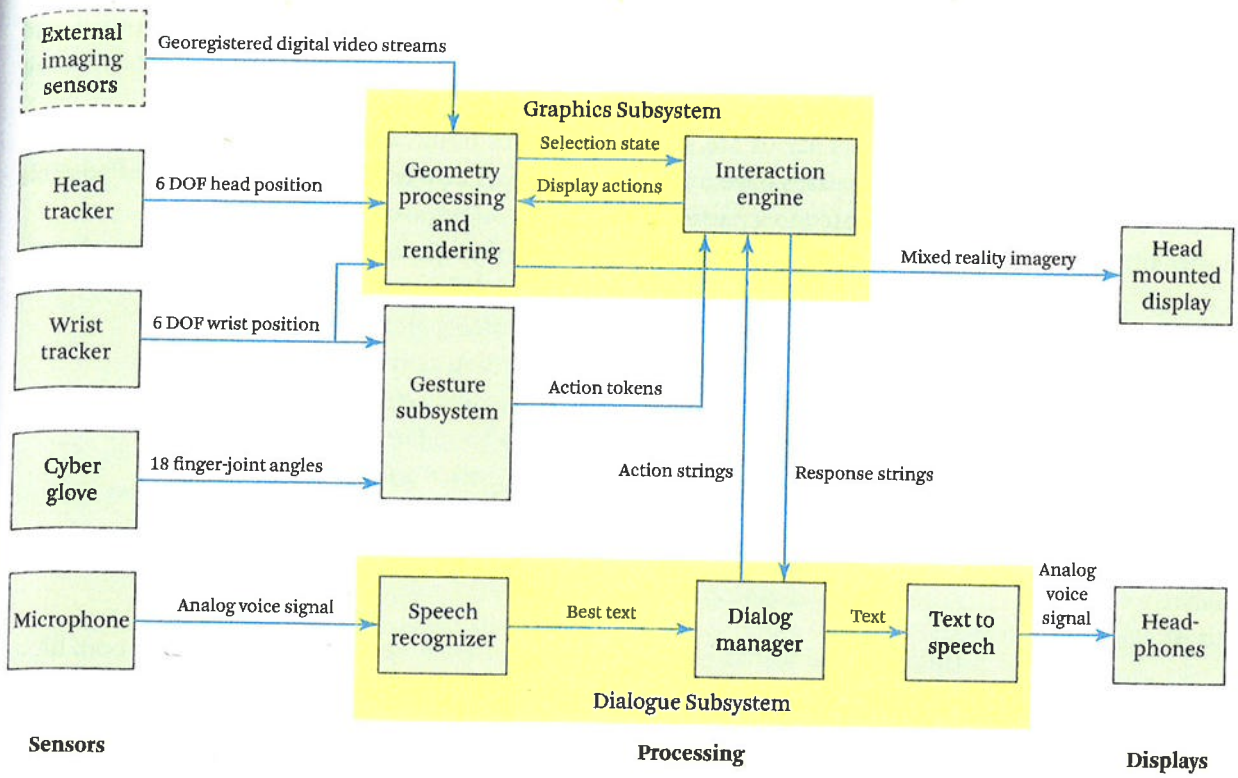


Figure 32.2 **A block diagram for a multimodal VR system developed at HRL Laboratories.** (Based on Neely et al. (2004))

32.2.2 Block Diagrams

Block diagrams are high-level diagrams showing interconnections between different system components. Boxes are used to denote components and arrows connecting the components show the input and output between the components. Block diagrams are used to show the overall relationship of components without concern for the details. Figure 32.2 shows a block diagram for a multimodal VR system with various software and hardware components.

32.2.3 Use Cases

A **use case** is a set of steps that helps to define interactions between the user and the system in order to achieve a goal. Defining use cases helps developers identify, clarify, and organize interactions in a way that makes it easier to implement the interactions. As interaction complexity grows, the need for explicit use cases increases.

Meta Platforms, Inc.
Exhibit 1067
Page 423

Use cases often start from user stories (Section 31.12), but are more formal and detailed enabling developers to get closer to implementation. Use cases can also result from task analysis (Section 32.1).

Use case scenarios are specific examples of interactions along a single path of a use case. Use cases have multiple scenarios, i.e., a use case is a collection of possible scenarios related to a particular goal.

There is no standard way of writing a use case. Some prefer visual diagrams whereas others prefer written text. Some form of the following should be included.

- Use case name
- Intended result
- Description
- Preconditions
- Primary scenario
- Alternate scenarios

Outlining the scenarios with hierarchical levels can be useful to provide both high-level and more detailed views. This also allows the filling in of the details as more specific detail is needed. Colors or different font types can be used to see what needs to be implemented and what has already been implemented/tested, the differences between the primary path and alternative paths, or who is assigned to what steps.

Use cases do not need to exist in isolation. A use case might extend upon an already existing use. Or a large use case might be broken apart into multiple use cases if there are common steps that are used multiple times throughout the larger use case. These smaller use cases can also be included within other larger use cases where appropriate. This helps to make user interactions consistent across the entire experience. For example a pointing selection technique might be a small use case that is part of a larger use case that requires many selections and manipulations. The same small pointing use case might also be used as part of an automated travel technique (select an object to travel toward and then the system somehow moves you to that location).

32.2.4 Classes

As the design specification evolves, it is necessary to get clearer on how the design will actually be implemented in code. This can be done by taking information from the previous steps and organizing that information into common structures and functionality into themes that can be described and implemented as classes. A **class** is a template for a set of data and methods. Data corresponds to nouns and properties

Meta Platforms, Inc.
Exhibit 1067
Page 424

from information collected in previous steps, and methods correspond to verbs and behaviors.

A **program object** is a specific instantiations of a class. A class exists by itself without having to be compiled into an executable program (e.g., classes exists even when the virtual environment does not yet exist) whereas an object is a representation of information organized in a manner defined by a class that is included in the executable program. An example of a program object is a perceivable virtual object in the virtual environment such as a rock on the ground that can be picked up.

A real-world analogy of a class is a set of blueprints for a house. Blueprints are like a class that describes the house but is not a house itself. Construction workers instantiate the blueprints into a real house. The blueprints can be used to build an arbitrary number of houses. Likewise, classes define what can be instantiated into objects inside a virtual environment. An example of a class is a box class that can be instantiated as a specific box that exists in the virtual environment. The box class can be instantiated many times to create multiple box objects in the environment. The class might define what characteristics are possible, and the box object can take on any of those possible characteristics. For example, different box objects in the environment might have different colors.

Classes and objects do not have to represent physical things. They might also represent properties, behaviors, or interaction techniques. A simple example is a color class. Instantiated objects of the color class might be specific colors such as red, green, or blue. These color objects, along with other objects, could then be associated to each individual box to describe what the box looks like.

A **class diagram** describes classes, along with their properties and methods, and the relationship between classes. Class diagrams directly state what needs to be (or already is) implemented. Figure 32.3 shows an example of such a diagram consisting of a single class.

32.2.5 Software Design Patterns

A software design pattern is a general reusable conceptual solution that is used to solve commonly occurring software architecture problems. They are described from a system architect's and programmers point of view, where implementation structure is described by relationships and interactions between classes and objects. Design patterns speed up development by using tested, proven concepts that have been useful for other developers, and provides a common language to communicate with other developers.

Some common patterns for VR are factories to instantiate objects (e.g., for the user to create many similar objects); adapters to connect different software libraries (e.g.,

Meta Platforms, Inc.
Exhibit 1067
Page 425

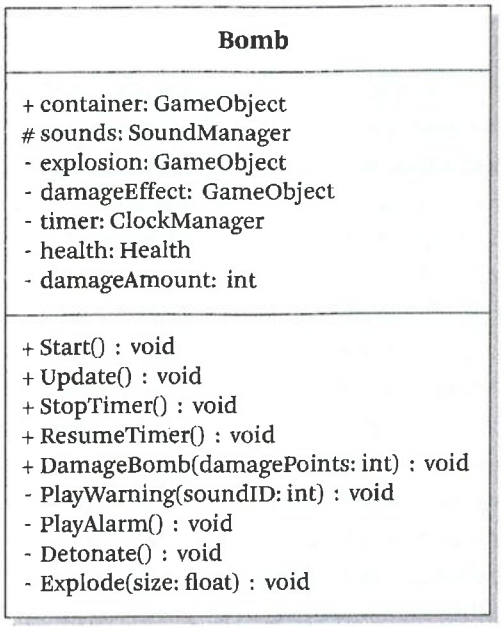


Figure 32.3 **An example class diagram.**

to create a single way of supporting different HMDs); singletons to ensure only one object of a type exists (e.g., a single scene graph or hierarchical structure of the world); decorators to dynamically add behaviors to objects (e.g., adding a sound to an object that previously did not support sound); and observers so objects can be informed when some event occurs (e.g., callbacks so some part of the system knows when the user has pushed a button).

32.3 System Considerations

32.3.1 System Trade-Offs and Making Decisions

Considering the different trade-offs of a project leads to a new understanding of the intricately interlocked interplay of factors (Brooks, 2010) and how they might be implemented. Clarity of understanding trade-offs help the team to choose hardware, interaction techniques, software design, and where to focus development. Such decisions depend upon many factors, such as hardware affordability/availability, time available to implement, interaction fidelity, representative user's susceptibility to sickness, etc. Table 32.1 shows some of the more common decisions that must be made when de-

Meta Platforms, Inc.
Exhibit 1067
Page 426

Table 32.1 **Some common VR decisions and example choices.**

Decision	Example Choices
Hand input hardware	None, Leap Motion, Sixense STEM
Number of participants	Single user, multi-player, massively multiplayer
Viewpoint control pattern(s)	Walking, steering, world-in-miniature, 3D multi-touch, automated
Selection pattern(s)	Hand selection, pointing, image-plane, volume-based
Manipulation pattern(s)	Direct hand manipulation, proxy
Realism	Real-world capture, cartoon world
Vection	None, short periods of self motion, linear motion only, active, passive
Physical space	Sitting, real walking
Intensity	Relaxing, heart-thumping
Sensory cues	Visual, auditory, haptics, motion platform
Posture	Sitting, standing, walk around

signing and implementing a VR experience. Most of these system decisions should be made early in the project. Even if it turns out the decision was wrong then it will quickly become apparent and can be changed accordingly at an early iteration.

In some cases it is appropriate to support multiple options. Be careful of trying to support too many options as supporting everything by implementing for the "average" results in no single optimal experience. Instead, implement different metaphors and interactions that most appropriately fit each option. It is generally better to start by choosing a single option and optimize for it before adding secondary options, features, and support.

32.3.2 Support of Different Hardware

Although some hardware has similar characteristics, other hardware can have very different characteristics (Chapter 27). Supporting different hardware that is in the same class can enable a wider audience since users may not all have the same access to the same hardware. Fortunately, in some cases the specific hardware only slightly changes the experience. For example, the Sony PlayStation Move is the same class of input device as the Sixense STEM Controller (both are 6 DoF tracked hand-held controllers). Supporting both will require additional code but the core interaction

Meta Platforms, Inc.
Exhibit 1067
Page 427

techniques can remain the same. Just because hardware is similar, do not assume testing on one piece of hardware will apply to all hardware in that same class; test across the different hardware and optimize as needed.

It can be dangerous to support different hardware classes for a single experience. Supporting different input device classes as being interchangeable is difficult and most often results in the experience being optimized for none of them. Those experiences that take full advantage of and are dependent upon unique characteristics of some input device class should not support other hardware input device classes. For example, the Sony Playstation Move and Sixense STEM are very different from Microsoft Kinect or Leap Motion. The exception is for special cases when the experience is designed to take advantage of multiple input device classes in a hybrid system where all users are expected to have access to and use all hardware.

If different hardware classes must be supported, then the core interaction techniques should be independently optimized for each in order to take advantage of their unique characteristics. A tracked hand-held controller will have different interactions and experiences than a bare-hands system.

32.3.3 Frame Rate and Latency

A frame rate that maintains at least the refresh rate of the HMD should be obtained from the beginning and maintained throughout the project (Section 31.15.3). Otherwise it will be more difficult to optimize at a later time (e.g., lower polygon-count assets may need to be recreated). Fortunately, assuming scene complexity is reasonable, today's hardware makes this relatively easy to achieve. Frame rate should be carefully observed as new assets are added and code complexity increases. Even occasional drops in frame rate can be uncomfortable to users. Consistent latency is as important as low latency (Section 15.2).

Some rendering algorithms/game engines perform multiple passes to add special effects. In some cases this can add one or more additional frames of latency even while a high frame rate is achieved. Content developers should not only rely on frame rate and should measure end-to-end latency with a latency meter (Section 15.5.1). As discussed in Section 18.7, prediction and warping can be used to reduce some of the negative effects of latency, but this only works well for latencies within a 30 ms range.

32.3.4 Sickness Guidelines

Developers should respect the adverse effects of VR (Part III) even if they are not prone to VR sickness themselves. Waiting for feedback from the Learn Stage can dramatically slow down the iterative process and might require complete redesign/reimplementation if problems are not take care of while implementing. It is as important for

Meta Platforms, Inc.
Exhibit 1067
Page 428

programmers to understand adverse health effects and how to mitigate them as much as anyone on the team.

32.3.5 Calibration

Proper calibration of the system is essential and tools should be created for both developers and users to easily and quickly calibrate (if not automated). Examples of calibration include interpupilary distance, lens distortion-correction parameters, and tracker-to-eye offset. If these settings are not correct, scene motion will occur while rotating the head, which will result in motion sickness. See Holloway (1997) for a detailed discussion of various sources of HMD calibration error and their effects.

32.4 Simulation

Simulation can be more challenging for VR than it is in more traditional applications. This section briefly discusses some of these challenges and some ways to solve for them.

32.4.1 Separate Simulation from Rendering

Simulations should be executed asynchronously from rendering (Bryson and Johan, 1996; Taylor et al., 2010). Slow updates of parts of the scene are ok as long as rendering resulting from head motion occurs at the HMD refresh rate and under 30 ms of latency. For example, a scientific simulation of colliding galaxies might execute remotely on a supercomputer with updates coming in only once per second. This slow dynamic updating of the data will be obvious to the user, but the user will still be able to view the static aspects of the data (e.g., the user can move his head into the non-moving data) in real time.

For realistic physics simulations, it is usually required that the simulation be calculated at a fast update rate. This is especially true if the physics simulation is being used to render haptic forces (Section 3.2.3) as haptics updated at a rate less than 1000 Hz can result in objects not feeling solid. Higher rates are even better at making objects feel even more solid (Salisbury et al., 2004).

32.4.2 Fighting Physics

Mixing human interaction with physics simulation can be extremely challenging in VR because the simulation of objects "fight" with where the physical hand is. This is due to the results of the physics simulation contradicting where the hand actually is. This can result in the object appearing to jitter (moving quickly back and forth) because the human input and the simulation "fight" or go back and forth between the calculated

Meta Platforms, Inc.
Exhibit 1067
Page 429

pose and the actual hand pose. The simple solution is to not fight. That is do not try to have two separate policies simultaneously determine where an objects is.

The most common solution when an object is picked up by a user is to stop applying physics simulation to that object. The object may still apply forces to other non-static objects (e.g., hitting a ball with a baseball bat). This lack of physics being applied to the hand-held object results in the object moving through static objects (e.g., a wall or large desk) since those static objects don't move and don't apply forces back to the hand-held object (although sensory substitution such as a vibrating controller can be applied as discussed in Section 26.8). Although not realistic, this approach is often less presence-breaking than the alternative of jitter and unrealistic physics.

The other option is to ignore hand position when a hand-held object penetrates another object. This works well when penetration is shallow but not for deeper penetrations (Section 26.8). Unfortunately penetration depth typically cannot be controlled (i.e., most VR systems do not provide physical constraints) so this is typically not a good option.

32.4.3 Jittering Objects

Physics simulation even without user intervention can cause objects to jitter. For example, an object will sometimes appear to jitter on the ground even when the user is not touching it. This can occur due to different reasons such as round-off error/numerical approximation or using linear estimations for nonlinear behavior. For example, the simulation might overestimate the distance an object has fallen resulting in the object penetrating the ground. The simulation then applies a force to move the ball back up just above the ground. But then gravity takes over and the object falls just under the surface and the cycle continues. A simple solution can be implemented by clamping object motion to zero when that motion reaches some minimum value. This unfortunately causes objects to sometimes appear to suddenly stop when the minimum value is reached. Allowing a small amount of penetration without causing a collision response can also help. This, however, can result in objects appearing to penetrate other objects if the value is too large. More elegant solutions can be implemented such as applying a stronger damping force when the minimum value is reached.

Solving for rag-doll character motions with purely physically-based simulation can also result in instability that appears as jitter. Thus, if a physically-based rag doll is used, the simulation might not control the character's bones directly but be a hybrid solution of the physics simulation feeding into code that simplifies the simulation results, appearing as smoother believable motions.

Meta Platforms, Inc.
Exhibit 1067
Page 430

32.4.4 Flying Objects

It is also often the case that mixing human input and simulation will result in extremely large forces causing objects to fly quickly off into the distance. This can happen when the physical hand pushes an object past the surface of some other object and then releases the object. The physics simulation then takes over and overcompensates by applying a large force to move the object instantaneously toward the surface. If the hand has penetrated the object too far then the force will be large resulting in the object flying off into the distance in an unrealistic manner. Specialized code might be required to detect such occurrences and then handle the situation differently than using the standard physics formulas. For example, if the object is released inside an object then the object might first be moved toward the closest surface before letting the physics engine take over.

Similarly, simulation instability can result when multiple physically-simulated objects are bound closely together. The instability can exacerbate quickly and cause objects to fly off into space. Such scenarios can be quite chaotic, and difficult to automatically detect and correct (David Collodi, personal communication, May 4, 2015). Thus, the designer should avoid having multiple physically-simulated objects interacting whenever possible, especially if the objects are in an enclosed space or are tightly constrained against one another (e.g., a stack of blocks in a small pit or multiple boxes joined together in a circular fashion).

32.5 Networked Environments

A networked VR system where multiple users share the same virtual space has enormous challenges beyond the challenges of a single-user system. Essentially, a collaborative VR system is a distributed real-time database with multiple users modifying it in real time with the expectation that the database is always the same for all users (Delaney et al., 2006a).

32.5.1 Ideals of Networked Environments

Network consistency is the ideal goal that at any point in time, all users should perceive the same shared information at the same time (Gautier et al., 1999). Unfortunately, perfect consistency is impossible to achieve due to technical challenges such as network latency, packet loss, and difficulty in maintaining true quality of service. However, by understanding the challenges and violations of network consistency, we can work toward the ideal of creating higher quality shared experiences.

Network consistency can be broken down into synchronization, causality, and concurrency (Delaney et al., 2006a).

Meta Platforms, Inc.
Exhibit 1067
Page 431

- **Synchronization** is the maintenance of consistent entity pose and timing of events for all users. Ideally, all clocks are synchronized.
- **Causality** (also known as ordering) maintains consistent ordering of events for all users. Ideally, all events are executed on each computer in the true order they occur no matter which computer they occurred on.
- **Concurrency** is the simultaneous execution of events by different users on the same objects. Object ownership/control must be resolved. Ideally, there are no conflicts over shared objects.

Violations of any of the three challenges listed above can lead to inconsistent states of the world between users and breaks in presence. Problems specific to networked environments include the following.

- **Divergence** is the temporal-spatial state of an object being different for different users. Divergent objects result in users interacting with those objects inconsistently, sometimes leading to inconsistent behavior causing object states to diverge even further. An example is simulation divergence that occurs when physics simulations are executed on different computers.
- **Causality violations** are events that are out of order so that effects appear to occur before cause. A ball appearing to bounce before a remote user appears to dislodge the ball is a causality violation.
- **Expectation violations** (also known as intention violations) are resulting effects from concurrent events that are different than the expected or intended effects. An expectation violation occurs when a remote user change the color of an object at approximately the same time as the local users changes the color of the same object. The local user expects it to turn one color but it changes to some other color.

In addition to being consistent, networked VR systems should also be easily useable and believable. Responsiveness and perceived continuity are two primary contributors of the user experience as it relates to building networked systems.

- **Responsiveness** is the time taken for the system to register and respond to a user action. Synchronization, causality, and concurrency could more easily be achieved by adding a delay for all users but would result in low responsiveness. Ideally, users can interact with all networked objects as if they were local without having to wait for confirmation from other computers.
- **Perceived continuity** is the perception that all objects behave in a believable manner without visible jitter or jumps in position and that audio sounds smooth.

Meta Platforms, Inc.
Exhibit 1067
Page 432

32.5.2 Message protocols

Communication between computers occurs through packets. A **packet** is a formatted piece of data that travels over a computer network. The type of packets used to convey information can have a big impact on performance and synchronization.

UDP (user datagram protocol) is a minimal connectionless transmission model that operates on a best-effort basis. Packets are simply sent out to a destination without any guarantee of delivery, ordering, or duplicated packets. Packets might not be received, for example if the receiving computer is overloaded with too many incoming packets. UDP should be used when low latency is a priority, updates occur often, and each update is not essential (e.g., when the state of the world will get updated soon enough if a recent packet was not received). Examples of when to use UDP are for continually updated character position and audio.

TCP (transport control protocol) is a bidirectional reliable ordered byte stream that comes at the cost of additional latency. TCP works by the receiving computer acknowledging with the sending computer that the packet was successfully received. Information is also included in each packet that guarantees the order that packets were received will match the order that they were sent. TCP should be used when state information is only sent once (or occasionally) to ensure the receiving computer can update its state of the world to match the sender's. An example of when to use TCP is when some single one-time event has occurred and the receiving computer needs to update its state of the world based on that one-time event.

Multicast is a one-to-many or many-to-many distribution of group communication where information is simultaneously sent to a group of addresses instead of a single address at a time. Clients simply subscribe to the multicast channel and then receive updates until they unsubscribe. Multicast works well in theory but unfortunately there are technical challenges that often keep multicast from working well and it can cause network overload depending how it is implemented on network hardware (that is often out of control of the developer). True network multicasting (also known as IP multicasting) is ideal but not all commercial routers support it (or it is disabled) so an application layer on top of the network is often built to simulate true multicasting. Unfortunately, the application layering approach is not as efficient. When possible, network multicasting should be used over application layering multicasting.

32.5.3 Network Architectures

There are many ways of connecting virtual worlds. However, network architectures are generally described as client-server, peer-to-peer, or hybrid architectures.

Peer-to-peer architectures transmit information directly between individual computers. Each computer maintains its own state of the world, and ideally matches all

Meta Platforms, Inc.
Exhibit 1067
Page 433

other computer states of the world. In a true peer-to-peer architecture, all peers have equal roles and responsibilities. The greatest advantage of peer-to-peer architectures for VR is fast responsiveness. One of the challenges of peer-to-peer architectures is discovering virtually local peers so that users can see each other. Thus, hybrid architectures are often used for networked virtual environments where some centralized server with global knowledge connects users.

Client-server architectures consist of each client first communicating with a server and then the server distributing information out to the clients. Fully authoritative servers control all the world state, simulation, and processing of input from clients. This makes all the users' worlds consistent at the cost of responsiveness and continuity; although this can be improved by the local system giving the illusion of responsiveness and continuity—see Section 32.5.4. As proven by Valve and others, authoritative client-server models can work well for a limited number of users (Valve, 2015). To scale to larger persistent worlds, multiple servers should be used to reduce computational and bandwidth requirements, as well as to provide redundancy if the server crashes.

Some network architectures do not clearly fall within either of the above architectures. **Hybrid architectures** use elements of peer-to-peer and client-server architectures.

Non-authoritative servers technically follow the client-server model but behave in many ways like a peer-to-peer model because the server only relays messages between clients (i.e., the server does not modify the messages or perform any extra processing of those messages). Primary advantages are ease of implementation, and knowledge of all other users. Disadvantages are that systems can become out of sync and clients can cheat as they are not governed by an authoritative system.

Super-peers are clients that also act as servers for regions of virtual space and can be changed between users (e.g., when a user logs off). The greatest advantage of super-peers over pure server-client architectures is increased heterogenity of resources (e.g., bandwidth, processing power) across peers. However, super-peer networks can be challenging to implement well.

Other hybrid architectures use peer-to-peer for transferring some data (e.g., voice) while also using a server to secure important data and to maintain a persistent world. Figure 32.4 shows an example of a VR system for distributed design review that uses a mix of a server (HIVE Server; Howard et al., 1998), audio multicast, and a peer-to-peer distributed persistent memory system (CAVERNSoft; Leigh et al., 1997).

32.5.4 Determinism and Local Estimation

Since bandwidth limits the rate at which packets can be sent and received, a straightforward implementation can result in remotely-controlled entities appearing to up-

Meta Platforms, Inc.
Exhibit 1067
Page 434

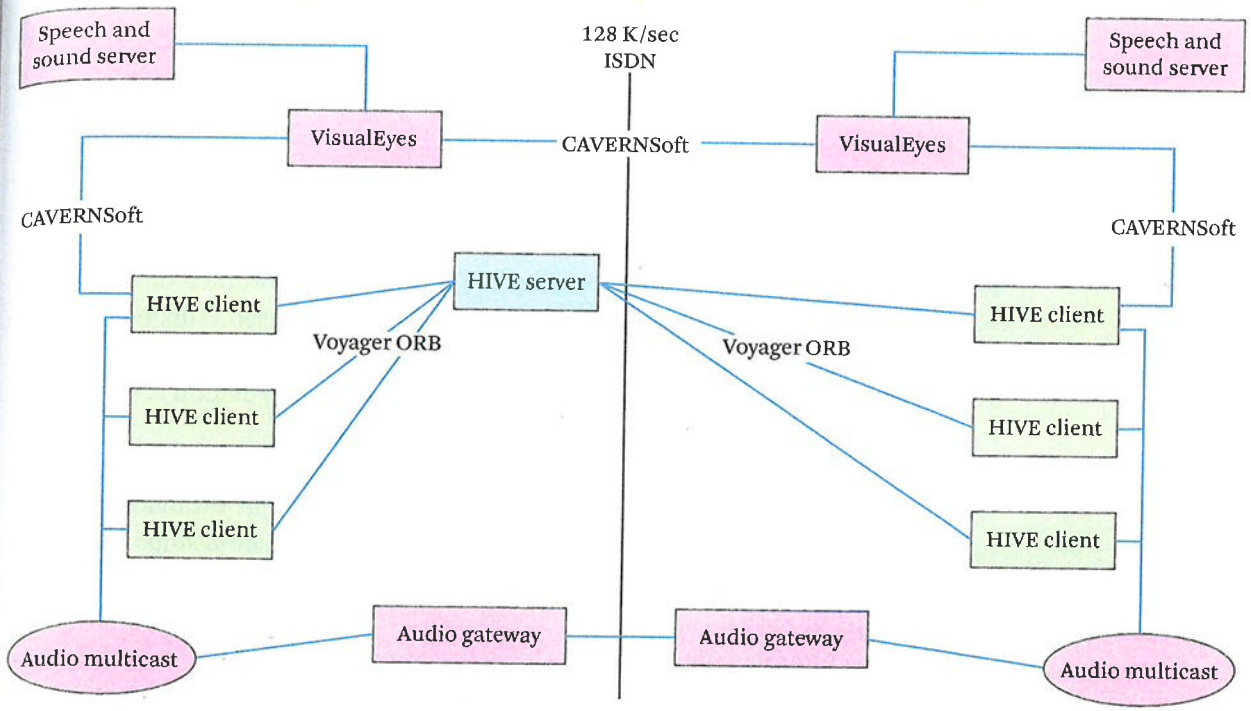


Figure 32.4 **Example of a network architecture with a centralized server and audio separated from other data.** (Based on Daily et al. (2000))

date in discrete steps—i.e., they will appear to be frozen for moments of time with frequent teleportation to new locations as new packets are received. Such discontinuities can be reduced depending if the actions are deterministic, nondeterministic, or partially deterministic. Nondeterministic actions occur through user interaction, such as when a user grabs an object, and remote nondeterministic actions can only be known by receiving a packet from the computer on which the action took place. Deterministic actions should be computed locally and there is no need to send out or receive packets in such cases.

Partially deterministic actions of remote entities (e.g., a user is steering in a general direction) can be estimated every rendered frame in order to create perceived continuity. **Dead reckoning** or extrapolation estimates where a dynamic entity is located based off of its state defined by information from its most recently received packet. As a new packet is received the state is updated with the new true information. If the estimated state and updated state diverge by too much, then the entity will appear to instantaneously teleport to the new location. Such discontinuities can be reduced

Meta Platforms, Inc.
Exhibit 1067
Page 435

by interpolating between the estimated position and the new position defined by the new packet. Valve's Source game engine uses both extrapolation and interpolation to provide a fast and smooth gaming experience (Valve, 2015). Although there is a trade-off of smoothness versus entity latency, this form of latency does not cause sickness as the local user's head tracking and frame rate is independent of other users.

32.5.5 Reducing Network Traffic

Reducing network traffic is often a necessity to optimize networked environments no matter what architecture or communication protocol is used. This is especially true when scaling up to a large number of users. In addition to general technique such as data compression, sending out packets only when information is needed can dramatically reduce network traffic (Delaney et al., 2006b).

One way to reduce network traffic is to compute the divergence between the local true state and the remote estimated state (e.g., entity position estimated by dead reckoning). For example, divergence will occur when a user starts/stops moving or steers in a different direction. **Divergence filtering** only sends out packets when the divergence of an entity reaches some threshold.

Relevance filtering only sends out a subset of information to each individual computer as a function of some criterion. A common way to do this is for individual servers to control some virtual space (e.g., a grid structure consisting of cells with an individual server controlling each cell) so that packets are only sent to nearby users. Server overloading caused by a large number of users in a small area can be a problem. This can be solved by using a dynamic grid/cell sizes that varies cell size depending on the number of users in an area (e.g., Hu et al., 2014).

Animations can also be used to reduce bandwidth requirements. For example, animating a user's legs is often more appropriate when a user moves instead of updating the actual joints of the avatar every frame based on incoming network packets (and legs are often animated anyway even in a single-user system). Gesture recognition can be used to first interpret a user's signal (e.g., a hand wave) so a high-level signal can be sent out and then reproduced on each client.

Audio often takes an order of magnitude or more bandwidth than other updates (Daily et al., 2000) and is especially a problem when many users attempt to speak simultaneously (Jesse Joudrey, personal communication, April 20, 2015). Stress tests can quickly be conducted by getting many users to speak simultaneously through the same server. Supporting spatialized audio requires each user's audio stream to remain independent, so combining all audio into a single stream on a server, as is often done with non-immersive communication, is not an option. If spatialized audio is used, it is better to send the audio using peer-to-peer rather than all audio going through a

Meta Platforms, Inc.
Exhibit 1067
Page 436

server. If feasible, multicasting should be used so that users automatically subscribe and unsubscribe from the audio streams of virtual neighbors. Relevance filtering can be used to only send audio to nearby users. Less bandwidth, and thus lower quality audio, can also be allocated to users who are further away or not in the general forward-looking direction (Fann et al., 2011). Depending on the level of realism intended, the application might also provide the option to subscribe to teammates or important announcements regardless of distance.

32.5.6 Simultaneous Interactions

Two or more users simultaneously interacting with the same object is extremely difficult to do in networked environments and should be avoided if at all possible. Only one user at a time should own an object so that no other users can interact with it. This can be done through a token. A **token** is a shared data structure that can only be owned by a single user at a time. Before a user interacts with an object he must first request the token to make sure another user does not own it. This can make interaction difficult as there will be a delayed response when attempting to interact with an object. One way to reduce this latency is to request the key when the user approaches an object, but before reaching for or selecting the object (Roberts and Sharkey, 1997).

32.5.7 Networking Physics

Networked physics simulations are even a greater challenge than physics simulation on a single computer (Section 32.3) because multiple representations of the same action occur at different physical locations in time and space. Theoretically, the same results should occur on different computers using the same equations and code. However, slight differences between clients (e.g., round-off error or differences in timing) cause the physics simulation to diverge over time. If a ball rolls in one direction on one computer it may roll in a different direction on a different computer, and eventually the ball may be in an entire different part of the environment. Because of this an authoritative server or single computer should own the simulation. Other computers can estimate the simulation, but the owning computer should send out regular updates to correct the current state, as described in Section 32.5.4.

32.6 Prototypes

A **prototype** is a simplistic implementation of what is trying to be accomplished without being over concerned with aesthetics or perfection. Prototypes are informing because they enable the team to observe and measure what users do, not just what

Meta Platforms, Inc.
Exhibit 1067
Page 437

they say. Prototypes win arguments and measurement using these prototypes trumps opinion.

A **minimal prototype** is built with the least amount of work necessary to receive meaningful feedback. Temporary tunnel vision is ok when trying to find an answer to a question about a specific task or small portion of an experience. At early stages, the team shouldn't bother debating the details. Instead focus on getting *something* working as quickly as possible and then modify and add additional functionality/features on top of that or start from scratch if necessary after learning from the basic prototype.

Have a clear goal for each prototype. Only build primitive functionality that is essential to achieving the goal and reject elements not essential to that goal. Non-critical details can be added and refined later. Give up on trying to look good and feeling that a prototype is ugly, unfinished, or not ready. Expect that it won't be done well the first time or the fifth time and to be confronted by many unforeseen difficulties for what at first may have seemed to be a relatively straightforward goal. Discovering such difficulties is what prototypes are for and it is best to find such problems as soon as possible. The sooner the bad ideas can be rejected and the best basic concepts and interaction techniques found, the sooner the limited resources can be placed on creating quality experiences.

Building prototypes quickly with minimal effort also adds the additional psychological benefit of not caring if the prototype is thrown away or not. Fast iteration and many prototypes result in an attitude of not being committed to one thing so developers don't feel like they are killing their babies.

Prototypes can vary significantly depending on what is trying to be accomplished. For example, a question about if a certain aspect of a new interaction technique will work or not is very different than determining if the general market likes an overall concept.

32.6.1 Forms of Prototypes

A **real-world prototype** does not use any digital technology whatsoever. Instead of using a VR system, team members or users act out roles. This might include physical props or real-world tools such as laser pointers. Advantages are that such prototypes can often be created and tested spontaneously. Some disadvantages are lack of controlling conditions, difficulty in capturing quantitative data, simulated actions may not match VR action, and feedback is limited to high-level structure/logic.

A **Wizard of Oz prototype** is a basic working VR application but a human "wizard" behind a curtain (or on the other side of the HMD) controls the response of the system in place of software. The Wizard typically enters commands on a keyboard or controller after the user verbally states his intention (e.g., to state where to travel to or to simulate a voice-recognition system). When done well this sort of prototype can

Meta Platforms, Inc.
Exhibit 1067
Page 438

be surprisingly compelling and the user may not even realize a human is controlling the system. Although collecting data for a specific implementation cannot be done with this type of prototype, high-level feedback can be obtained.

Programmer prototypes are prototypes created by and evaluated by the same programmer or team of programmers. Programmers continually immerse themselves in their own systems to quickly check modifications to their code. This is how most programmers naturally operate, sometimes conducting hundreds of mini experiments (e.g., change a variable value or logic statement to see if it works as expected) in a day.

Team prototypes are built for those not directly implementing the system. Feedback from others is extremely valuable as it is often difficult to evaluate the overall experience when working closely on a problem. This can also reduce group think and having the belief that your VR experience is the best in the world when it is not. Team prototypes are used for quality assurance testing. Note team might include those external to the core team such as when performing expert evaluations (Section 33.3.6) via consultants. Other teams that understand the concepts of prototypes also work well.

Stakeholder prototypes are semi-polished prototypes that are taken more seriously and typically focus on the overall experience. Often stakeholders are less familiar with VR than they like to admit. They expect a higher level of fidelity that is closer to the real product, which helps them to more fully understand what the final product will be like. Most stakeholders, however, want to see progress, so don't wait until the product is finalized. Their feedback about market demand, understanding business needs, competition, etc. can be extremely valuable.

Representative user prototypes are prototypes designed for feedback that should be built and shown to users as soon as possible. Users that have never experienced VR before are ideal for testing for VR sickness since they have not had the chance to adapt. These prototypes are primarily used for collecting data, as described in Chapter 33. The focus should be on building a prototype that best collects data for what is being targeted.

Marketing prototypes are built to attract positive attention to the company/project and are most often shown at meetups, conferences, trade shows, or made publically available via download. They may not convey the entire experience but should be polished. A secondary advantage of these prototypes outside of marketing is that a lot of users can provide feedback in a short amount of time.

32.7 Final Production

Final production starts once the design and features have been finalized so that the team can focus on polishing the deliverable. When final production begins, it is time to stop exploring possibilities and stop adding more features. Save those ideas for a

Meta Platforms, Inc.
Exhibit 1067
Page 439

later deliverable. One of the great challenges of developing VR applications is that when stakeholders experience a solid demo, they get excited and start imagining possibilities and requesting new features. Whereas the team should always be open to input, they should also be aware and make it clear that there is a limit of what is possible within the time and budget constraints. Some suggestions might be easy to add (e.g., changing the color of an object), but others add risk as modifying a seemingly simple part of the system can affect other parts. Adding features also takes away from other aspects of the project. The team should make it clear that if stakeholders want additional features, there are trade-offs involved and the new features may come at the cost of delaying other features.

32.8 Delivery

Deployment ranges from the extremes of making the application continuously available online with weekly updates to multiple team members traveling to a site (or multiple sites) for a full system installation.

32.8.1 Demos

Demos are the lifeblood of VR. After all, most people are not interested in a conglomeration of documentation, plans, and reports—they want to experience the results. They are also not interested in excuses of how it was working at the last conference or how something is broken. It is the VR experience that gives confidence to others that progress is being made and the team is creating something of value.

Conferences, meetups, and other events give the team their time to show off their creation. Scheduling demos are also a great way to create real accountability to get things done and get them done well. In the worst case, the team will be significantly motivated to do better next time after a demo turns out to be a disaster.

Before traveling, set the demo up in a room different from where equipment is normally located to make sure no equipment is missed when packing. If the equipment is beyond the basics, then take the packed material to yet a different room and set up again to ensure everything that is needed was indeed packed. Then pack two back-ups of everything. If at all possible, set up in space where the demo will take place at least one day in advance to make sure there will be no surprises. On the day of the demo, arrive early and confirm everything is working.

Being prepared for onsite demos is also essential. With all the excitement of what is possible with VR, you never know when a stakeholder or celebrity is going to stop by wanting to see what you have. Note this does not mean anyone wanting a demo should be able to walk any time of day to test the system—it simply means to be prepared

Meta Platforms, Inc.
Exhibit 1067
Page 440

when the important people show up. The team's time is valuable and demos can be distracting so demos should be given sparingly unless of direct value to the project. A single individual should be responsible for maintaining and updating onsite demos and equipment so when someone important shows up the system is more than barely working. The same individual might be responsible for keeping a schedule of demo events, or that might be some other individual.

32.8.2 Onsite Installation

Full project delivery often includes onsite installation. In such cases, do not assume installation will go smoothly. Debugging, calibration, development, and extra electrical/gaffer tape should be expected for complicated installation with multiple pieces of hardware. Issues might include electromagnetic interference and incompatible connections/cables with a client's system. In addition, training of staff may be required to use and maintain the system. What is obvious to you may not be obvious to them. Support and updates might also be included as part of a contract.

32.8.3 Continuous Delivery

The opposite of onsite installation is continuous online delivery where updates are provided often. Agile methods recommend delivering a new build that goes out to customers on a weekly basis. This has value in forcing the team to focus on what's important and to be accountable, as well as provides plenty of opportunity for data collection (A/B testing of changing one thing for a subset of customers is most common in this situation—see Section 33.4.1).

Meta Platforms, Inc.
Exhibit 1067
Page 441

The Learn Stage

All life is an experiment. The more experiments you make the better.
—Ralph Waldo Emerson

The **Learn Stage** is all about continuous discovery of what works well and what doesn't work so well. The pursuit of learning about one's creations and how others experience it is more essential for VR than for any other technology.

Learning about what works in VR and how to improve upon one's specific application can take many forms. Learning might be as simple as changing values in code and immediately seeing the results, which is not uncommon for programmers to do hundreds of time in a single day. Or learning might take months to conduct due to formal experiment design, development, testing hundreds of users, and analysis. Although not as rapid as code testing, this section leans toward fast feedback, data collection, and experimentation. This enables the team to learn quickly how well or not ideas meet goals, and to correct course immediately as necessary. It is also highly recommended to utilize VR experts, subject-matter experts, usability experts, experiment-design experts, and statisticians to ensure you are doing the right things to maximize learning. The earlier problems are found the less expensive they are to fix. Finding a problem near final deployment can result in failure for the entire project.

Understanding the learning/research process not only enables one to conduct his own research but also enables one to understand the research of others, where it might apply to one's own needs or project, and to intelligently question claims instead of blindly accepting them.

Like the other stages of iterative design, the Learn Stage will evolve over the course of the project. Initial learning will consist of informally demonstrating the system to teammates to get general feedback. Eventually more sophisticated methods of collecting and analyzing data might be employed that very well may go beyond what is

Meta Platforms, Inc.
Exhibit 1067
Page 442

discussed here. Readers are encouraged to use only the concepts that most directly apply to their project; all projects will not use all concepts. However, readers will benefit by being aware of all of them.

33.1 Communication and Attitude

Effective communication is essential for learning. This includes communication between team members as well as communication with users. Effective learning and communication is heavily dependent on attitude of how one seeks out and reacts to constructive criticism. Those that have problems with receiving criticism about their work will not make it far with VR. A positive mindset and attitude about continual improvement is essential, and often the greatest breakthroughs occur through breakdowns that result from others pointing out issues that are not obvious to those creating the experience.

When demonstrating systems and asking for feedback, think positively for your own self, team members, and users even when failure occurs. Consider failure to be learning experiences and do not fear failure. Of course the goal is success, but often we don't really know why we succeeded. With an appropriate attitude toward failure, it is often possible to figure out why, to ensure that the problem will not happen again.

When communicating with users, the following points are essential in order to maximize learning.

- Do not blame/belittle users or their opinions/interactions with the VR application. Doing so shuts them down emotionally where they will not effectively provide feedback.
- Actively investigate difficulties to determine how the project can improve.
- Assume what others are doing is partially correct and then provide suggestions that enables them to correct and move on.
- When someone points out a problem you are already aware of, thank them for noticing, and encourage them to continue looking for problems.
- Instead of using words like failure, talk about learning.

As an additional benefit of communicating effectively with users, those giving feedback will also feel good about themselves and a sense of contribution and ownership which can turn them into fans/evangelists when the product becomes publically available. Such community involvement is essential for crowdfunding campaigns (e.g., Kickstarter).

Meta Platforms, Inc.
Exhibit 1067
Page 443

33.1.1 VR Creators are Unique Users

A common mistake content creators, especially programmers, make when creating VR experiences is that they assume what works for themselves will work for everyone else. Unfortunately, this is especially not the case for VR. Programmers often use the system in very specific ways without trying all unanticipated actions. Programmers also may have adapted to sensory incongruencies where they do not get sick as non-adapted users do.

Programmers don't typically believe strongly that user's opinions don't matter—in fact, they are often aware they should be collecting feedback but see it as an inconvenience to regularly communicate with users; it is just not a priority. Most programmers do enjoy showing off their work occasionally, but such demos are not done frequently enough, are not done in a manner that results in quality data collection, and do not result in changes because opinions are not taken seriously enough.

Programmers' time is extremely valuable, and it is true programmers should not be constantly bothered with giving demos and collecting data. Thus, the answer is that other team members should be used to collect data from users. However, it should be the case that programmers do at least occasionally participate in data collection, so that they take the feedback more seriously. There is greater resistance to seeing or hearing a report stating what one has created doesn't work well, compared to observing in person how it doesn't work well.

33.2 Research Concepts

Research takes many forms and has many definitions. One definition of **research** is a systematic method of gaining new information and a persistent effort to think straight (Gliner et al., 2009).

How many VR teams conduct research with the intent to truly understand what works for representative users? That is how many VR professionals collect external feedback apart from giving the standard VR demo in the office, meetup, or conference? Unfortunately, very few. This is a reason why there are normally so many unknowns in creating an engaging VR experience, leaving the success of the project to chance instead of finding and pursuing the optimal path toward success.

This section covers background concepts that are useful for designing, conducting, and interpreting research in an effective manner.

33.2.1 Data Collection

Data collection enables the effectiveness of VR aspects to be specified, quantified, and compared. Data collection often involves a combination of both automated data

Meta Platforms, Inc.
Exhibit 1067
Page 444

capture and observation by human raters. Some commonly collected data for VR include time to completion, accuracy, achievements/accomplishments, frequency of occurrence, resources used, space/distance, errors, body/tracker motion, latency, breaks in presence, collisions with walls and other geometry, physiological measures, preferences, etc. Clearly, there are many measures that can be useful for VR. The team should collect different types of data rather than relying on a single measure.

Data can be divided into qualitative or quantitative data. Both are important as they each provide unique insight about a design's strength and weaknesses.

Quantitative data (also known as numerical data) is information about quantity—information that can be directly measured and expressed numerically. Quantitative data is best gathered with tools (e.g., a questionnaire, a simple physical measuring device, a computer, or body trackers) that require relatively little training and result in reliable information. Quantitative data analysis involves various methods for summarizing, comparing, and assigning meaning to data, which are usually numeric and which usually involve calculation of statistical measures. Typically, experimental approaches (Section 33.4) focus on collecting quantitative data.

Qualitative data is more subjective—it is biased and can be interpreted differently by different people. Such data includes interpretations, feelings, attitudes, and opinions. The data is often collected through words (whether spoken or written) and/or observation. Qualitative data is not directly measured with numbers but, in addition to being summarized and interpreted, can be categorized and coded. Constructivist approaches (Section 33.3) rely heavily upon qualitative data.

Both quantitative and qualitative approaches are useful for developing better VR experiences. In fact, both types of data are often collected from the same individuals. Qualitative data is often gathered first to get a high-level understanding of what might be later studied more objectively with quantitative data. Even for teams focusing only on qualitative data, it is useful to have a basic understanding of the core concepts of quantitative research, which can help to understand when qualitative data and its conclusions might be biased.

33.2.2 Reliability

Reliability is the extent to which an experiment, test, or measure consistently yields the same result under similar conditions. Reliability is important because if a result is only found once, then it may have happened by chance. If similar results occur with different users, sessions, times, experimenters, and luck then we can be highly confident there must be some consistent characteristic of what is being tested.

Measures are never perfectly reliable. Two factors that affect reliability are stable characteristics and unstable characteristics (Murphy and Davidshofer, 2005). **Stable**

Meta Platforms, Inc.
Exhibit 1067
Page 445

characteristics are factors that improve reliability. Ideally, the thing being measured will be stable. For example, an interaction technique is stable if different users have a consistent level of performance on a task when using that technique. **Unstable characteristics** are factors that detract from reliability. These include a user's state (health, fatigue, motivation, emotional state, clarity and comprehension of instructions, fluctuations of memory/attention, personality, etc.) and device/environment properties (operating temperature, precision, reading errors, etc).

An **observed score** is a value obtained from a single measurement and is a function of both stable characteristics and unstable characteristics. The **true score** is the value that would be obtained if the average was taken from an infinite number of measurements. The true score can never be exactly known, although a large number of measurements provide a good estimate. When testing participants repeatedly, to measure learning effects for example, it is important to have high reliability so that it is more likely the observed score was a result from learning versus occurring by chance.

Reliability values are often expressed as a correlation coefficient (Gliner et al., 2009) that takes a value between -1 and 1, as explained in Section 33.5.2.

33.2.3 Validity

A reliable measure may consistently measure something, but it might be measuring the wrong thing. **Validity** is the extent to which a concept, measurement, or conclusion is well founded and corresponds to real application. Validity is important to understand when conducting research to make sure one is actually measuring and comparing what one thinks they are measuring and comparing, conclusions are legitimate, and the results apply to the domain of interest.

Overall, validity is a function of many factors and researchers further break validity down into more specific types of validity. Unfortunately, researchers cannot agree on how to organize and name the different types of validity. Below general validity is broken down into face validity, construct validity, internal validity, statistical conclusion validity, and external validity.

Face Validity

Face validity is the general intuitive and subjective impression that a concept, measure, or conclusion seems legitimate. Face validity is one of the first things researchers consider when collecting data.

Construct Validity

Construct validity is the degree to which a measurement actually measures the conceptual variable (the construct) being assessed. A measure has high construct validity

Meta Platforms, Inc.
Exhibit 1067
Page 446

if it captures the hypothetical quality that it claims to be measuring. More specifically, a measure with high construct validity adequately covers all aspects of the construct; correlates with other measures of the same construct; is not affected by variables not related to the construct; and accurately predicts concurrent and future measurements, behavior, or performance. A common example of construct validity is the ability of an assessment tool to differentiate between experts and novices performing a given task.

The goal of collecting data may be to measure user performance, user preference, some aspect of a generalized interaction technique, or a specific implementation for analyzing usability issues. At early stages of testing, bugs and imperfections will be present. Because of system imperfection, the researcher must be careful what is being measured. In many cases, the specific implementation is not what is of interest, because problems are known that will be fixed. Rather, questions are more often about general user preferences, performance, and perception that are unrelated to bugs already on the maker's task list to be fixed. If a system artifact has been measured when that is not the intention, then a violation of construct validity has occurred. Researchers must be careful of concluding that a concept is not effective in general just because it does not work in its current implementation.

Internal Validity

Internal validity is the degree to which a relationship is causal. Internal validity depends on the strength or soundness of the experiment design and influences whether one can conclude that the independent variable or intervention caused the dependent variable to change. When the effects of the independent variable cannot be separated from the possible effects of extraneous circumstances, the effects are said to be confounded (Section 33.4.1). Internal validity can be threatened or compromised by many confounds including the following.

History is an extraneous environmental event that occurs when something other than the manipulated variable happens between the pre-test and the post-test, An example is if data collection is conducted on different days—some world event may have occurred that altered opinion or motivation for performing well.

Maturation is personal change in participants over time. The changes might be due to changes that are not relevant to that being tested. For example, participants may grow tired, become sick, or become resistant to VR sickness.

Instrumentation occurs due to change in the measurement tool or observers' rating behavior. For example, a tool for measuring reaction time or physiological measures could go out of calibration. Observers that rate participants change their criteria over time, or worse different observers are used. Calibration, providing guidelines, and training observers are important for reducing instrumentation

Meta Platforms, Inc.
Exhibit 1067
Page 447

bias. The room setting, hardware setup, and software should not change between sessions.

Selection bias occurs due to non-random assignment or self-selection of participants to groups. Selection bias is best solved by randomly assigning participants to groups. If participants self-select themselves into groups then bias can be reduced, but not removed, by not letting them know of the differing conditions. Fans volunteering to test a system will certainly be biased toward liking a VR experience more so than the general targeted population.

Attrition (also known as mortality) is the dropout of participants. This is a problem for validity when the attrition for one group is different from another group. This is a major problem when participants drop out due to VR sickness. If one condition causes one group to have more sickness, then that group will lose more participants. Those that do not drop out will likely have a higher tolerance for sickness and thus the results may be due some characteristic that correlates to tolerance of sickness rather than the factor being studied.

Retesting bias often occurs when the data is collected from the same individual two or more times. Participants are likely to improve/learn on the first attempt due to practice that carries over into later tests. Results may be due to this practice rather than due to the manipulated variable.

Statistical regression occurs when participants are selected in advance based on high or low scores. Many of those who scored low or high scored that way due to chance. On subsequent testing, scores tend to average out due to the initial scores being different from their true average score. There is no way to determine if the changes occurred due to statistical regression or due to the factor being tested. Selection of participants should not be based off of previous scores unless multiple tests showed consistent and reliable results.

Demand Characteristics are cues that enable participants to guess the hypothesis. Participants might modify their behavior (consciously or subconsciously) to look good, appear politically correct, or to make the experimenter look good or bad.

Placebo Effects occur when a participant's *expectation* that the experiment causes a result instead of the experimental condition itself causing the result. For example, pre-tests with the Kennedy Simulator Sickness Questionnaire (Section 16.1) have been found to cause heightened sickness as reported in post-tests (Young et al., 2007). Or, if the participant believes that one interaction technique is better than another technique, then it may be the result occurs to the belief rather than the mechanics of the interaction.

Meta Platforms, Inc.
Exhibit 1067
Page 448

Experimenter Bias occurs when the experimenter treats participants differently or behaves differently. This might be explicit bias due to the experimenter encouraging the participant to perform better under one condition, but is more often unintentional. The Experimenter may intend to not be biased, but may not even realize his own body language and tone of voice convey a preference for one condition. Ways to minimize experimenter bias are to make the experimenter blind to the conditions (e.g., randomize conditions and do not allow the experimenter to see what the participant sees), use naive experimenters (i.e., hire an assistant that would not know the difference of conditions even if he saw the conditions), and/or to minimize the experimenter's time with the participant (e.g., have the instructions given and data collected by the computer).

Statistical Conclusion Validity

Statistical conclusion validity is the degree to which conclusions about data are statistically correct. Some common threats to statistical conclusion validity include the following.

A false positive is a finding that occurs by chance when in fact no difference actually exists. (e.g., there is a chance that flipping an unweighted coin lands heads-up several times in a row). If the experiment is conducted again then the difference will not likely be found.

Low statistical power most commonly occurs due to not having a large enough sample size. A common mistake researchers make is to claim there is no difference between conditions, when in actuality there may be a difference but the researchers did not find the difference due to not having enough statistical power. In fact, an experiment can never prove two things are truly identical (an experimental result shows they are likely different with some probability), although techniques do exist that can show two things are close within some defined range of values.

Violated assumptions of data are wrongly assumed properties of data. Different statistical tests have different assumptions. For example, many tests assume data is normally distributed (Section 33.5.2). If these assumptions are not valid then the experiment may lead to wrong conclusions.

Fishing (also known as data mining) is the searching of data via many different hypotheses. If a difference is found when many variables are explored, the findings may be due to random chance. For example, if 100 different types of coins are flipped 10 times each then one or more of those coin types may land heads

Meta Platforms, Inc.
Exhibit 1067
Page 449

up 10 out of 10 times just by chance—not necessarily because that coin type is weighted. Different statistical corrections can be used to compensate for the increased likelihood of finding a difference in multiple tests.

External Validity

External validity is the degree to which the results of a study can be generalized to other settings, other users, other times, and other designs/implementations. The results that one interaction technique is better than some other technique might only apply to the specific configuration of the lab, the type of users, the specifics of the VR system (e.g., field of view, degrees of freedom, latency, software configurations). Some VR research results from the 1990s does not necessarily apply to today due to different users (e.g., researchers versus consumers) and due to changes in technology. At that time, latency of 100 ms or more was regarded as acceptable. Today, such numbers are completely unacceptable. Today's researchers have to use their best judgment on a case-by-case basis to decide if they want to take into account results from a previous time or reconduct some of those experiments.

If something changes in the design or implementation of the system then the previously reliable findings may no longer apply. For example, one interaction technique might be superior to another interaction technique for the system being tested but not necessarily for a system with different hardware. Ideally, experimental results will be robust to different forms of hardware, different versions of design and software, and across some range of conditions. In many cases, experiments should be conducted again when hardware has changed or significant design changes have occurred. Similar results will give confidence that the design is more robust, possibly even to specific conditions not yet tested.

Of course different techniques will never generalize to all situations and all hardware. Different input devices can have very different characteristics as discussed in Chapter 27. Minimal computer requirements (e.g., CPU speed and graphics card capabilities) should also be determined based off of testing. This is much more important than traditional desktop and mobile systems due to risk of sickness (Part III). Even for results that have high external validity, VR applications should never be shipped that have not been extensively tested with the targeted hardware.

33.2.4 Sensitivity

Measures may be consistent (reliable) and truly reflect the construct of interest (validity) but may lack sufficient sensitivity to be of use. **Sensitivity** refers to the capability of a measure to adequately discriminate between two things; it is the ability of an experimental method to accurately detect an effect, when one does exist. Is the measure

Meta Platforms, Inc.
Exhibit 1067
Page 450

capable of distinguishing between multiple levels of the independent variable? If not, then find a way to increase sensitivity.

33.3 Constructivist Approaches

Constructivist approaches construct understanding, meaning, knowledge, and ideas through experience and reflection upon those experiences rather than trying to measure absolute or objective truths about the world. The approach focuses more on qualitative data and emphasizes the integrated whole and the context that data is collected in. Research questions using this approach are often open-ended but structured enough to be useful for summary and analysis.

This section describes various methods of collecting data in order to better understand VR experiences and to improve upon those experiences.

33.3.1 Mini-Retrospectives

Mini-retrospectives are short focused discussions where the team discusses what is going well and what needs improvement. Such retrospectives should be kept short but performed often. It is much better to have weekly 30 minute mini-retrospectives rather than monthly half-day retrospectives.

Retrospectives should be constructive. Jonathan Rasmusson states in The Agile Samurai (Rasmusson, 2010) the retrospective prime directive:

> Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand. In other words it's not a witch hunt.

Ideally, mini-retrospectives will result in themes for future iterations and areas to track that the team wants to improve upon.

33.3.2 Demos

Demos are the most common form of acquiring feedback from users. Demos are a great start for getting a general feel of others' interest. Demos should be given often to stay in close communication with the intended audience, to understand real users, to receive fresh ideas, and to market the project. Most importantly, demos give something for the team to work toward and to show progress is being made.

However, the team should distinguish between giving demos and collecting data. Whereas demos are easy to do and do provide some basic feedback, they are the least useful method of collecting data unless they are combined with more data-focused approaches. Most often the point of demos is to market the project. Because of this,

Meta Platforms, Inc.
Exhibit 1067
Page 451

data collected during demos is typically not quality data. This is due to many factors such as data collection not being a priority, users often not providing honest opinions (i.e., they are polite), feedback almost always being remembered rather than recorded, the chaos of many users, answering questions instead of collecting data, and bias of those giving the demo and reporting the results. When data collection really is a goal of a demo, then structured interviews and questionnaires can be used to collect less biased data.

The best learning from a demo occurs when a public demo goes horribly wrong in a big way. This type of learning can be extremely painful. But if the company/team can survive the disaster, then such a lesson usually ignites a team into taking massive action to build something of quality and to make sure to do things well next time so they will not be humiliated again.

33.3.3 Interviews

Interviews are a series of questions asked by a real person and are more flexible and easier for the user than questionnaires. An interviews results in the best data when conducted immediately after the person being interviewed experienced the VR application. If the interview cannot be performed in person then it can be performed over telephone or (worse) in text form.

Interviews are different from testimonials. The intention of a testimonial is to produce marketing material showing that real people like something. The intention of an interview is to learn. Interviews guidelines should be created in advance that are designed to keep the interview on track and to reduce bias.

Appendix B provides an example of a simple interview guideline document where the intention is to generally improve a VR experience. When the intent is to improve upon more specific aspects of a VR experience, then more specific questions should be included.

Listed below are some tips for getting quality information out of interviewees.

Establish rapport. Skilled interviewers get to know and quickly become friends with the interviewees before asking questions specific to the intent. Differences in personal style, ethnicity, gender, or age may reduce rapport resulting in less truthful answers. For example, a professor or other authority figure conducting the interviewer may result in some interviewees not answering with their true thoughts. Attempt to match the interviewer with the intended audience based on the personas created, as described in Section 31.11.

Perform in a natural setting. Conducting interviews in unnatural or intimidating settings such as a lab can result in little response from interviewees. Many game

Meta Platforms, Inc.
Exhibit 1067
Page 452

labs have rooms set up as living rooms. The team can also travel to the site of the interviewee instead of having the interviewee travel to the side of the interviewers.

Time. Don't spend more than 30 minutes per demo/interview maximum in a single session. Set time limits by scheduling interviews back to back. It is better to interview multiple individuals than to interview one individual in depth. Individuals can be followed up with if they have more useful things to say.

33.3.4 Questionnaires

Questionnaires are written questions to which participants are asked to respond to in writing or on a computer. Questionnaires are easier to administer and provide for more private responses. Note questionnaires are different than surveys as the intent for surveys is to make inferences about an entire population, which requires careful sampling procedures. Surveys are typically not appropriate for evaluation of VR applications.

Asking for background information can be useful for determining if participants fit the target population that resembles personas created in the Define Stage (Section 31.11). Conversely, such information can be used to better define personas if participants are self-selected. The information can also be used to make sure a breadth of users is providing feedback, to match participants when comparing performance between participants, and to look for correlations of experience with performance.

The Kennedy Simulator Sickness Questionnaire as described in Section 16.1 is an example of a questionnaire commonly used for VR studies.

Close-ended questions provide options that are selected by circling or checking

Likert scales contain statements about a particular topic, and participants indicate their level of agreement from strongly disagree to strongly agree. The options are balanced so there are an equal number of positive and negative positions.

Partially open-ended questions provide several options that participants can circle or check but also provide an option to fill in information if they don't feel the listed options are appropriate.

Open-ended questions are extremely useful for obtaining information about the experience that is not expected. However, some participants must be given extra encouragement to fill out such questions as it takes more cognitive effort than simply circling an option or checking a box.

An example questionnaire, used in evaluation of Sixense's MakeVR application (Jerald et al., 2013), is included in Appendix A. The questionnaire includes the Kennedy Simulator Sickness Questionnaire, a Likert Scale, a background/experience questionnare, and open-ended questions.

Meta Platforms, Inc.
Exhibit 1067
Page 453

33.3.5 Focus Groups

Focus groups are similar to interviews but occur in group settings. Group settings are more efficient than individual interviews and can stimulate thinking as participants build off of each other's ideas. Traditionally, focus groups have been used to determine how customers will perceive and react to new products or to gather opinions for improving political campaigns. For VR, participants typically use a prototype demo to base their feedback on. The goal is to stimulate thinking and elicit ideas aimed at improving on existing approaches and creating new experiences and interactions. Focus groups are extremely useful in the early phases of VR design as the feedback can be used to explore new concepts, better refine questions, and improve upon data collection via more structured approaches.

An excellent example of an effective focus group was conducted by Digital ArtForms as part of the National Institute of Health grant "Motion-Controlled Gaming for Neuroscience Education" (Mlyniec, 2013). Two focus groups consisted of two separate classrooms of 5th graders. After subject-matter expert Ann M. Peiffer, PhD, of Wake Forest School of Medicine described the basic concepts of stroke and its causes to the class, each child individually put together a virtual 3D puzzle of a brain that consisted of the different brain lobes. As they grabbed a virtual brain lobe with hand-held tracked controllers, audio and video embedded within the game explained what that lobe controlled. In the next level, a virtual character complained about and acted out a stroke symptom. The child then "cured" the character by injecting "medicine" into the appropriate lobe. Time constraints and scores were awarded to motivate students to complete the task.

In addition to collecting data via pre- and post-tests to determine learning effectiveness, the team asked the kids how the game might be improved and solicited ideas for future games. The children came up with the concept of zombies with symptoms caused by stroke. As the zombies would come at the player, the student would have to quickly determine what a zombie's deficit was and then reach for the correct brain lobe to throw at the zombie, curing it with the appropriate "medicine" before it could reach the player. This game concept would have never been conceived of without input from the children. Digital ArtForms is now developing a VR title as a result of the focus group.

33.3.6 Expert Evaluations

Expert evaluations are systematic approaches conducted by experts that identify usability problems from the user's perspective with the intent to improve the user experience. When done properly, expert evaluations are the most efficient method of improving upon the usability of the system.

Meta Platforms, Inc.
Exhibit 1067
Page 454

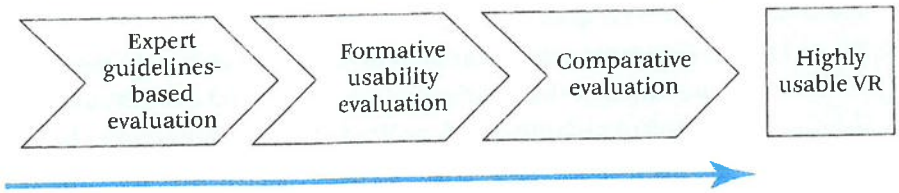


Figure 33.1 **Progression of expert evaluation methods.**

This section gives an overview of the methods that VR usability experts have found to be most efficient and effective for collecting quality data, enabling fast iteration toward ideal solutions. This is best done with different types of evaluations in succession as shown in Figure 33.1. Each evaluation component generates information that feeds into the next evaluation component, resulting in an efficient and cost-effective strategy for designing, assessing, and improving VR experiences (Gabbard, 2014). These methods are typically used for multiple aspects of the system and performed at different phases of the project.

Expert Guidelines-Based Evaluation

Expert guidelines-based evaluation (also known as heuristic evaluation) identifies potential usability problems by comparing interactions (either existing or evolving) to established guidelines (Gabbard, 2014). The identified problems are then used for recommendations for improving the design. This should be done early in the development cycle so issues can be corrected before affecting other aspects of the design.

Ideally, multiple VR usability experts perform independent evaluations of a prototype. Multiple evaluators results in quality and quantity of data that outweighs the cost of those evaluators. As this is for general VR usability most efficiently evaluated by experts, no representative users are involved in this type of evaluation. Evaluations should give a score of each issue's severity and describe why it is a problem. After each expert independently evaluates the application, results are combined and ranked to prioritize fixing critical usability problems and to design subsequent formative evaluations.

Unfortunately, traditional 2D interface guidelines are not appropriate for VR and no well-established guidelines specific to VR yet exist. Although not intended specifically for this type of evaluation, many of the guidelines in this book from Part III and Part V can be used. Other guidelines include the Oculus Best Practices Guide (*Oculus Best*

Meta Platforms, Inc.
Exhibit 1067
Page 455

Practices, 2015), Joseph Gabbard's taxonomy of usability characteristics (Gabbard, 1997), and NextGen Interactions internal documentation.

Formative Usability Evaluation

After expert guidelines-based evaluation has revealed and solved as many usability problems as possible, it is time to move on to formative usability evaluation. **Formative usability evaluation** diagnoses problems by gathering critical empirical evidence from an application and is applied during formative and evolving stages of design. The goal is to assess, refine, and improve usability, learning, performance, and exploration through observation of representative users (Gabbard, 2014; Hartson and Pyla, 2012). This improves the application by iteratively identifying problems throughout the design/development process so that the team can continually refine tasks and fine-tune interactions. The method should be conducted by VR usability experts as it relies heavily on a solid understanding of VR interaction that is dependent on context (i.e., it consists of more than a predefined list of items to evaluate). If done well, such evaluation can be quite efficient and effective for improving the usability of VR interactions.

Figure 33.2 shows the formative usability evaluation cycle. User task scenarios (which can be taken from task analysis—see Section 32.1) are first created to exploit and explore all identified tasks, information, and workflows. As users interact with the system they "think aloud" by explicitly verbalizing their actions, thoughts, actions, and goals (Hartson and Pyla, 2012). Evaluators collect qualitative and quantitative data in order to identify problems as well as strengths. This data is then analyzed and summarized with suggestions for improvements and emphasis on keeping features that work well. This information can also be used to improve user task scenarios for the next iteration of the formative evaluation.

One of the most useful results of evaluation is a list of critical incidents (Hartson and Pyla, 2012). A **critical incident** is an event that has a significant impact, either positive or negative, on user's task performance and/or satisfaction. This can include system crashes/errors, inability to complete a task, user confusion/disorientation, loss of tracking, sudden onset of discomfort, etc. Such critical incidents can dramatically impede usability and can have an effect on perception of application quality, usefulness, and reputation. Thus, identifying and correcting these problems as early as possible is essential to creating a quality experience.

Other data to collect includes accuracy and precision, task time to completion, number of errors, performance, and accomplishment of learning objectives. Such data can be measured through a scoring system visible to users, designed both to increase motivation and to target data collection.

Meta Platforms, Inc.
Exhibit 1067
Page 456

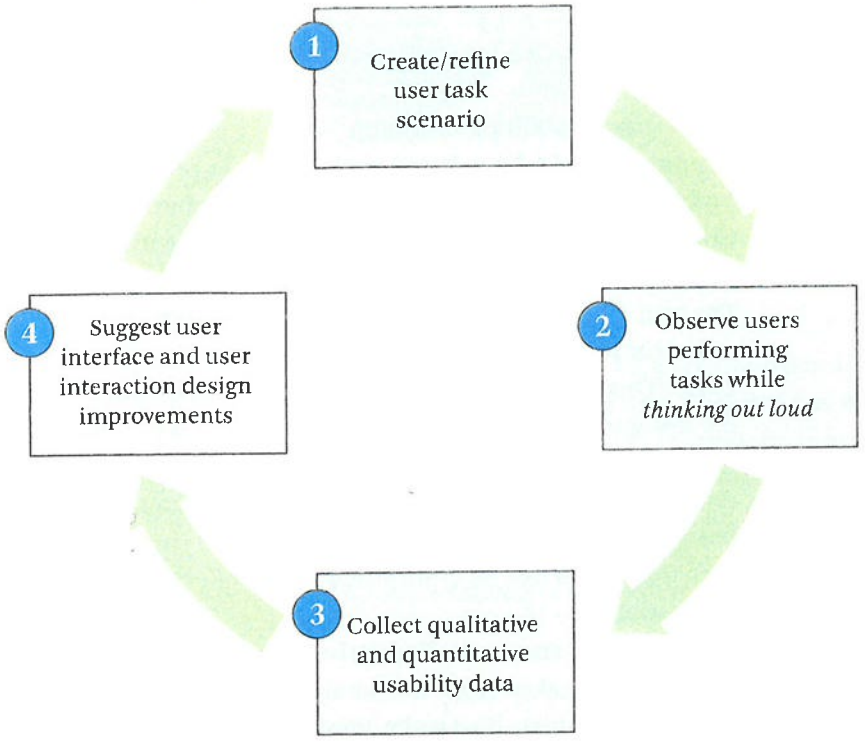


Figure 33.2 **Formative usability evaluation cycle.** (Based on Gabbard (2014))

During the final stages of formative evaluation, evaluators should only observe and not suggest how to interact with the system. This almost always uncovers problems of learning a system when no human is present to teach how to interact. Most users of the final experience will not have someone available to explain how to interact. Thus signifiers, instructions, and tutorials often result from this evaluation.

Comparative Evaluation

Comparative evaluation (also known as summative evaluation) is a comparison of two or more well-formed complete or near-complete systems, applications, methods, or interaction techniques to determine which is more useful and/or more cost effective. Comparative evaluation requires a consistent set of user tasks (borrowed or refined from a formative usability evaluation) that can be used to collect quantitative data to compare across conditions.

Evaluating multiple independent variables can help to determining the strengths and weaknesses of different techniques. Evaluators compare the best of a few refined

Meta Platforms, Inc.
Exhibit 1067
Page 457

designs by having representative users perform tasks to determine which of the designs is best suited for integration within a larger project or for delivery to customers. Comparative evaluation is also used to summarize how a new system compares to a previously used system (e.g., does a VR training system result in more productivity than a traditional training system?). However, be careful of threats to validity (Section 33.2.3) that may result in erroneous conclusions.

33.3.7 After Action Reviews

An **after action review** is a user debriefing with an emphasis on the user's specific actions. A member of the team discusses with the user his actions in order to determine what happened, why it happened, and how it could be done better. It is useful to watch and discuss real-world video of the user performing tasks along with a first person perspective, which can easily be accomplished by using screen/video capture technology recorded during the VR experience. If the software can support it, then observing the action from a third person viewpoint that can be interactively controlled also enables looking at the action from different perspectives.

33.4 The Scientific Method

The **scientific method** is an ongoing iterative process based on observation and experimentation. It typically begins with observations that lead to questions. These questions are then turned into testable predictions called hypotheses. These hypotheses can be tested in a variety of ways, including making further observations. The strongest tests come from carefully controlled and replicated experiments that gather empirical data. In order to gain confidence in a test result, the result should be replicated multiple times. The results of experiments more often leads to more questions than answers. Depending on the results of the tests, the hypothesis may require refinement, alteration, expansion, or rejection. This cycle of predict, test, and refine can repeat many times.

An **experiment** is a systematic investigation to answer questions and to gain new understanding. In most cases experimentation for the sake of creating VR applications is less formal then the extensive research methods performed by researchers more interested in scientific inquiry than creating experiences. Rigorous scientific/academic experiments are often overkill for those creating VR applications, and such formal experiments are not the focus of this book. However, understanding basic concepts of formal experiment can be useful for applied VR as such understanding can help design more basic informal experiments. Even if the perfect experiment cannot be conducted (and it rarely is perfect, even with months of planning), at least the team

Meta Platforms, Inc.
Exhibit 1067
Page 458

is aware of some of the pitfalls that might occur. Understanding formal experimental approaches also helps to comprehend research papers and interpret/explain results.

33.4.1 Overview of Experimental Design

This section describes the basics of a single iteration of the scientific method.

Explore the problem

The first step of exploring a problem is to obtain an understanding of what is to be studied. This can be done by learning from others (via trying their applications, reading research reports/papers, and talking with them); by trying oneself and observing others interacting with existing VR applications; by the constructivist approaches previously discussed; and by the various concepts discussed in the Define and Make Stages.

Formulate Questions

Once familiar with the overall problem, start by asking and answering the following questions:

- What specifically is the team trying to learn?
- What are the signals that provide feedback if something is working or not?
- What is the fastest and most effective way to get these questions answered?

State the Hypothesis

Answering the above questions, along with documentation from previous processes such as the listed assumptions and requirements, will lead to more specific questions that can be turned into hypotheses. A **hypothesis** is a predictive statement about the relationship between two variables. An example of a hypothesis is "Interaction technique X results in less time to complete task Y than interaction technique Z."

Until the team has gained extensive experience with basic experimental designs (or the team already contains a senior researcher with experience in conducting experiments), experiments should only compare two things at a time. Adding more complexity adds significant risk by potentially making wrong assumption, selecting the wrong statistical tests, having to add many more participants/sessions, misinterpreting results, etc.

Determine the Variables

To conduct an experiment, different variables must first be precisely defined.

The **independent variable** is the input that is varied or manipulated by the experimenter. An example of an independent variable is the selection interaction tech-

Meta Platforms, Inc.
Exhibit 1067
Page 459

nique used (e.g., object snapping versus precision mode pointing described in Section 28.1.2).

The **dependent variable** is the output or response that is measured. The dependent variable values resulting from the manipulation of the independent variable are what will be statistically compared to determine if there is a difference. Examples of dependent variables are time to completion for a task, reaction times, and navigation performance.

Confounding factors are variables other than the independent variable that may affect the dependent variable and can lead to a distortion of the relationship between the independent and dependent variables. For example, if one group of participants are studied in the morning and another group studied in the evening, then one group may perform differently due to being more tired than the other group, not necessarily because the thing being studied causes different performance.

Being aware of and considering threats to internal validity (Section 33.2.3) can help to find potential confounding factors. Once such confounding factors are determined to be a problem, they can often be removed by setting those factors to be constant. **Control variables** (also known as constant variables) are kept constant in an experiment in order to keep that variable from affecting the dependent variable.

Decide on Within Subjects or Between Subjects

A decision that must be made for all experiments is whether to design for within subjects or between subjects. Both have their advantages and disadvantages.

A **within subjects design** (also called repeated measures) has every participant experience each condition. Within subjects designs have the advantage of fewer subjects being required due to each participant experiencing all conditions and less variability within individuals. This results in more efficient data collection due to spending less time on recruiting, scheduling, training, etc. The disadvantage of within subjects design is that differences between conditions may be due to carryover effects instead of due to what is intended to be measured. **Carryover effects** occur when experiencing one condition causes retesting bias in a later condition. Examples are learning/training, fatigue, sickness, etc. One way to reduce bias from carryover effects is to counterbalance the order that participants experience the different conditions (e.g., half of the participants first experience condition A then B, half of the participants first experience B then A).

A **between subjects design** (also known as A/B testing when only two variables are being compared) has each participant only experience a single condition. Advantages are short times for participants, low dropout rate, and no carryover effects. The

Meta Platforms, Inc.
Exhibit 1067
Page 460

main disadvantage is the requirement to have a larger number of participants due to different participants needed for different conditions and larger variability between individuals.

Conduct Pilot Studies

A full experiment can be quite costly. If the experiment contains flaws or wrong assumptions then the entire study may be invalidated. A **pilot study** is a small-scale preliminary experiment that acts as a test run for a more full-scale experiment. Such preliminary studies are used to help determine feasibility, discover unexpected challenges, improve upon the experiment design, reduce time and cost, and estimate effect size (used to estimate the sample size needed to find a result).

Conduct the Experiment

Once researchers are confident the experimental design is solid, then they proceed on to full data collection from all participants. Note experimenters should not change conditions once data collection has started as this can cause confounding factors and nullify the results of the experiment.

Analyze the Data, Draw Conclusions, and Iterate

After all data has been collected, then statistical analysis (Section 33.5.2) as defined in the experimental design is performed. If the design has been precisely defined then there should be no ambiguity if a result was found or not. However, the experimenters likely discovered ways the experiment could be improved upon and gained insight for new questions to further explore and experiment with. The proces continues in an iterative manner.

33.4.2 True Experiments versus Quasi-Experiments

True experiments use random assignment of participants to groups in order to remove major threats to internal validity. Because of the random assignments, the only difference between the two groups of participants will be due to chance. True experiments are optimal for determining cause-effect relationships.

A quasi-experiment lacks true random assignment of participants to conditions. An example of not randomly assigning participants to groups is having participants self-select which group they will belong to (even if they don't know the difference of conditions between the two groups). Because of this non-random assignment, quasi-experiments are more likely to contain confounding factors. As a result it is more difficult to argue a cause-effect relationship. Quasi-experiments do have the advantage

Meta Platforms, Inc.
Exhibit 1067
Page 461

of being easier to set up than a true experiment when it is difficult to randomly assign participants.

33.5 Data Analysis

33.5.1 Making Sense of the Data

Data interpretation is not always obvious. Data collection almost always lead to further questions. In some cases, data will be contradictory due to differences in conditions, users, and the way the data is collected.

Look for Patterns

Single measurements rarely give the entire picture. Interpreting data from a single user is risky as variance between users can be large. Instead look for patterns over multiple users. Be careful of assuming finding a pattern is the truth as fishing for patterns can lead to false conclusions (Section 33.2.3).

Pay Attention to Outliers

An outlier is an observation that is atypical from other observations. Don't automatically disregard outliers without trying to understand first why the outlier occurred. Did the outlier occur because there was an error in the system? If so, this could be the most important signal telling you to fix the error! Sometimes these outliers can provide more insight than more typical data.

Verify Through Different Conditions and Signals

Realize the results of an individual experiment are rarely the generalized truth (Section 33.2.3). As the design, implementation, and hardware changes, consideration should be given about how such changes may change the experiment results. If the experiment was for a very specific situation then vary the conditions and run the experiment again.

33.5.2 Statistical Concepts

Whereas this book does not go into the details of statistical analysis, this section does cover the basics so all team members can have a basic understanding of statistics; have a common language to speak; spot wrong assumptions; reduce the odds of incorrectly summarizing or interpreting data; and better understand VR research of others and their technical papers. At least one member of the team should be good at statistical

Meta Platforms, Inc.
Exhibit 1067
Page 462

analysis. This can be a professional statistician but is more often a programmer or psychologist that is good at math and has experience using statistical software tools.

Measurement Types

Variables have different measurement scale types and interpreting data can depend on the type properties. Variable scale types are divided into categorical, ordinal, interval, and ratio variables.

Categorical variables (also known as nominal variables) are the most basic form of measurement where each value takes on a mutually exclusive label or name. Nominal values do not have an impied order or value. An example nominal variable is gender (male or female).

Ordinal variables are ordered ranks of mutually exclusive categories but intervals between the ranks are not equal. An example of an ordinal variable is obtained by asking the question "How many times have you used VR? A) Never, B) 1-10 times, C) 11-100 times, or D) over 100 times."

Interval variables are ordered and equally spaced, i.e., categories have equal value between categories. Interval variables do not have an inherent absolute zero. Temperature in Farenheit or Celsius and time of day are examples of interval variables.

Ratio variables are like interval variables but also have an inherent absolute zero. This enables meaningful fractions or ratios. Examples of ratio variables are number of users, number of breaks in presence, and time to completion.

Table 33.1 shows the ways values can be mathematically manipulated and statistically summarized depending on what measurement type is used. Be careful not to perform inappropriate calculations on data that are not ratio variables. For example, a common mistake is to multiply and divide interval data or to add/subtract ordinal data.

Table 33.1 Appropriate statistical calculations depend on the type of variable being measured. Checkmarks indicate it is appropriate to use the mathematical calculation with the measurement type.

	Categorical	Ordinal	Interval	Ratio
Counts and percentage	✓	✓	✓	✓
Median, mode, interquartile range		✓	✓	✓
Addition & subtraction			✓	✓
Mean & standard deviation			✓	✓
Multiply & divide				✓

Meta Platforms, Inc.
Exhibit 1067
Page 463

Descriptive Statistics

Descriptive statistics summarize the main features of a collection of information. Described below are some of the most common and useful concepts of descriptive statistics.

Averages. An **average** represents the middle of a dataset that the data tends toward. The average can take on different values depending on how it is computed. The three types of averages are the mean, median, and mode.

The **mean** takes into account and weighs all values of a dataset equally. The mean is the most common form of the average. Means can be overly influenced by an extremely small or large value that rarely occurs (i.e., an outlier). The mean is not appropriate when the data is categorical or ordinal.

The **median** is the middle value in an ordered dataset. The median is often more appropriate than the mean when data is skewed toward higher or lower values or there is one or more outliers. For example, the mean of the values (1, 1, 2, 3, 100) is 21.4 whereas the median is 2. In this case, the median 2 might be considered the more appropriate average of representing the central tendency of the data. The median is also most appropriate when the data is ordinal.

The **mode** is the most frequently occurring value in a dataset. The mode is most appropriate when the data is categorical.

Distribution of Data. A **histogram** is a graphical representation of a distribution of data. Figure 33.3 shows an example histogram where participants responded by choosing from an option of 7 discrete responses. In the case that measurement are not discrete, values can first be placed into bins (a series of intervals) and then the counts within each bin are displayed.

For some data, the histogram approaches what is called a normal distribution as the number of samples increases (Figure 33.4). A **normal distribution** (also known as a bell curve or Gaussian function) varies in width and height but is well defined mathematically and has useful properties for data analysis. Many statistical tests assume data is normally distributed.

In any experiment, there will always be variations in the collected data. The **spread** (also called variability) of a dataset signifies how dispersed the data is. Measures of spread include the range, the interquartile range, the mean deviation, the variance, and the standard deviation. All of these measures are always zero or greater, where a value of zero signifies all values within the dataset are identical. The larger the spread, then the more the data is spread out.

Meta Platforms, Inc.
Exhibit 1067
Page 464

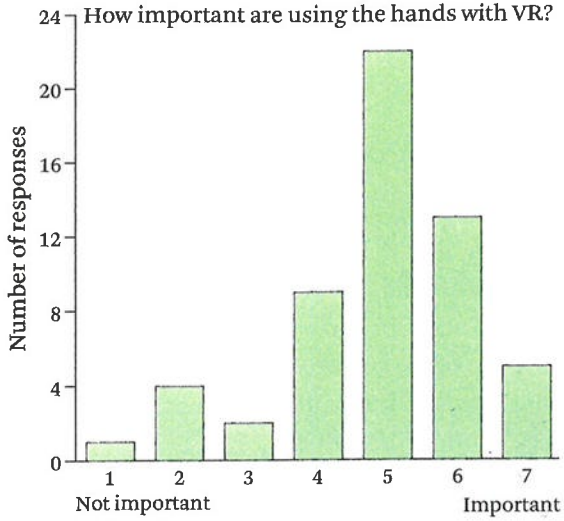


Figure 33.3 **A histogram of responses from a questionnaire to VR experts inquiring about the importance of using the hands with VR.** (Image courtesy of NextGen Interactions)

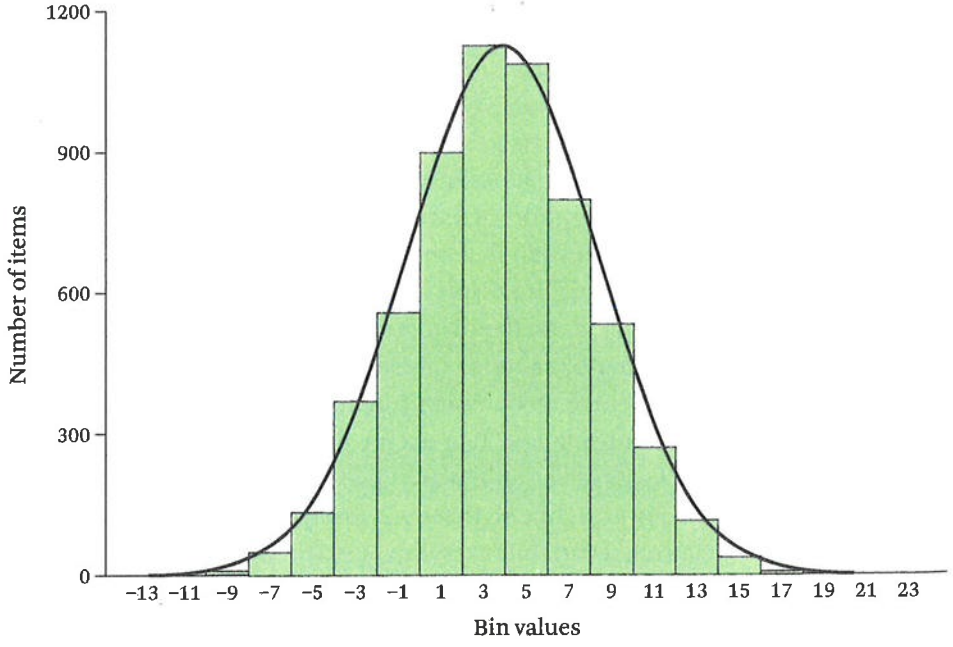


Figure 33.4 **A dataset wth an approximately normal distribution of data.**

Meta Platforms, Inc.
Exhibit 1067
Page 465

The **range** is simply the minimum and maximum values of a dataset. The **interquartile range** is the range of the middle 50% of the data. The interquartile range is often preferred to the total range since it automatically removes outliers and gives a better idea of where most of the data is located.

The **mean deviation** (also known as the mean absolute deviation) is how far values are, on average, from the overall dataset mean. The **variance** is similar, but is the mean squared distances from the mean. The **standard deviation** is the square root of the variance and is the most common measure of spread. The standard deviation is useful for several reasons, one being that it is in units of the original data (like the mean deviation). Another benefit is, assuming the data is normally distributed, plus or minus one standard deviation from the mean contains 68% of the data and plus or minus two standard deviations contain 95% of the data. For example, a VR training task might have taken on average 47 seconds to complete with a standard deviation of plus or minus 8 seconds, i.e., 68% of users completed the task between 39 and 55 seconds and 95% of users completed the task within 31 and 63 seconds.

Correlation

A **correlation** is the degree to which two or more attributes or measurements tend to vary together. Correlation values range between −1 and 1. A correlation equal to zero means no correlation exists. A correlation of 1 is a perfect positive correlation—as one value goes up the other value always goes up. Likewise, a negative correlation exists if as one value goes up the other values always goes down. Values between −1 and 1 signify as one value changes the other value tends to change, but not always. It is important to note that correlation is not sufficient for proving causation. There could be other reasons the data correlates.

Statistical and Practical Significance

Statistical significance states that with some confidence (a 95% confidence level is normally used) the statistical result of an experiment did not occur by chance and thus there is some underlying cause for the occurrence. If a coin was flipped 10 times and it came up heads every time, we can be fairly certain there was some event or property of the coin that caused it to turn up heads. However, there is a possibility this occurred by chance no matter how many times the coin was flipped and it came up heads. The p-value of an experiment represents the probability that a result occurred by chance rather than because there is something inherently truthful about the result. A p-value less than 0.05 (corresponding to a 95% confidence level) is typically considered to be statistically significant.

Meta Platforms, Inc.
Exhibit 1067
Page 466

Statistical significance does not necessarily mean the result is important enough to influence the next iteration of the project. **Practical significance** (also called clinical significance) implies the results are important enough that the results matter. For example, time to completion for a VR task might be statistically shorter for some condition by 0.1 seconds. However, if the average time to completion is ten minutes then the 0.1 second time improvement is probably not of practical significance and not worth being concerned about. If the 0.1 second improvement is a reduction in system latency from 0.15 second then it is certainly of practical significance.

Meta Platforms, Inc.
Exhibit 1067
Page 467

Iterative Design: Design Guidelines

Designing iteratively is more important for VR than for any other medium due to there being so many unknowns. Do not expect to find or create a single overall detailed VR design process that works across all types of VR projects. Instead, learn the different processes to know when they are most appropriate to use. Focus on continuous iteration of defining the project, making the application, and learning from users.

Philosophy of Iterative Design (Chapter 30)

Human-Centered Design (Section 30.2)

- Focus on the user experience.
- Give up traditional measures of software development such as lines of code.

Continuous Discovery through Iteration (Section 30.3)

- Give up on knowing and rationalizing everything from the start.
- Collect feedback as quickly and as often as possible from experts and representative users.
- Fail early and fail often in order to learn as quickly and as much as possible.
- If failure is not occurring often in early iterations, then attempt to innovate more.
- Always be on the lookout for changes in assumptions, technology, and business. Then change course as necessary.
- Step into the role of the user and try a broad range of VR experiences.
- There is more value in getting a bad prototype working than spending days thinking, considering, and debating.

Meta Platforms, Inc.
Exhibit 1067
Page 468

- Create a culture of accepting failure, especially early in the project to create a feeling that it is safe to experiment.
- Regard feedback and measurement from real application above opinion based on theory.

There Is No One Way—Processes are Project Dependent (Section 30.4)

- Prioritize team members and communication between team members over sticking to a formal process.
- Prioritize responding to change over following a plan.
- If it is not clear what processes to go with then just pick some and go with them. Then iterate on improving them or change the processes.
- Do not become attached to processes. Be willing to throw them away if they are not working.

Teams (Section 30.5)

- Keep teams small to maximize communication.
- If the team starts to become large then break into smaller teams.
- Make sure everyone is on board with the team philosophy.
- All team members should co-create, not just critique.
- Create a culture of learning from other team members, external suggestions from those experienced with VR, other disciplines, and user feedback.
- If any team member is close minded to ideas that are not his own, warn him and if he doesn't change remove him from the project as soon as possible.
- Collect input from a wide range of opinions and viewpoints but do not make decisions by committee.
- Assign an authoritative director that listens well but acts as a final decision maker on high-level items.
- If it is too difficult for a single person (the director) to understand how everything is connected then the design is overcomplicated. Simplify the design.

The Define Stage (Chapter 31)

- Do not spend too much time on the Define Stage in the first iteration.
- Be careful of analysis paralysis of trying to figure everything out before pulling the trigger.

Meta Platforms, Inc.
Exhibit 1067
Page 469

- If not sure about moving on to the Make Stage then move on to the Make Stage. Return to the Define Stage at a later time.
- In addition to documenting *what* decisions are made, document *why* decisions are made.

The Vision (Section 31.1)

- If some aspect of the project is unknown then guess. Articulated guesses beat unspoken or vague models.

Questions (Section 31.2)

- Get input from multiple individuals including all teammates, stakeholders, representative users, other teams, business developers, marketers, and expert consultants.
- Don't expect others to straight out tell you what they want. Help them discover what they want by asking questions.
- Ask questions that start with the word "why".
- Remember customers don't want an HMD, they want an experience and/or results.
- Understand more than just the VR aspect of the project. Understand the background and context of the project.

Assessment and Feasibility (Section 31.3)

- Do not assume VR is the right solution for every problem.
- Consider having the project consist of a mix of both VR and the real world.

High-Level Design Considerations (Section 31.4)

- Decide if you are focusing on design *with*, *for*, or *of* virtual environments.

Objectives (Section 31.5)

- Focus on benefits and outcomes instead of features.
- When creating objectives, use the SMART acronym—Specific, Measureable, Attainable, Relevant, and Time-bound.

Key Players (Section 31.6)

- All key players should be in agreement with the vision, really care about the project, and be committed to its success.

Meta Platforms, Inc.
Exhibit 1067
Page 470

- Understand many people may not have an interest no matter how good of a fit you think they might be. Listen to what they have to say and move on. Do not waste time chasing such individuals.

Time and Costs (Section 31.7)

- When negotiating contracts, separate out initial assessment and feasibility from implementation.
- Consider using milestones where additional payment is only received after achieving agreed upon milestones.
- Remember Brooks' Law—adding manpower to a late software project makes it later.
- Play planning poker to estimate development effort.
- Understand developers tend to overestimate their abilities even when they know they overestimate their abilities.

Risks (Section 31.8)

- Explicitly identify risk to bring awareness of any danger that might affect the project so that appropriate action can be taken to mitigate that risk.
- Focus on minimizing controllable risk.
- To reduce risk, make projects as short as possible. If a project is large break it down into smaller subprojects.

Assumptions (Section 31.9)

- Explicitly look for and declare assumptions so all team members begin from a common starting point.
- Be bold and precise in listing assumptions even when not sure. Wrong assumptions can be tested, vague assumptions cannot.
- Prioritize assumptions to test the riskiest assumptions first.

Project Constraints (Section 31.10)

- Make a single individual responsible for tracking and controlling constraints, with transparency to the entire team.
- Explicitly list constraints to narrow the design space.

Meta Platforms, Inc.
Exhibit 1067
Page 471

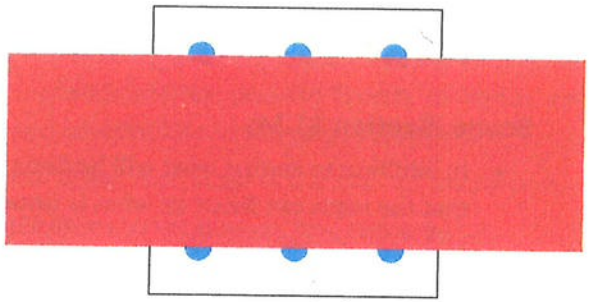


Figure 34.1 **A solution to Figure 31.7 of using only a single line to cover all nine dots. The problem never stated a limit to the width of the line.**

- Divide constraints into real constraints, resource constraints, obsolete constraints, misperceived constraints, indirect constraints, and intentional artificial constraints.
- Think outside the box by listing all misperceived constraints. Figure 34.1 shows a solution to Figure 31.8.

Personas (Section 31.11)

- Making generalizations to all people makes building a VR experience difficult. Target personas to make the design easier.
- Validate and modify personas on later iterations as more is learned about those actually using the system.
- For applications where personas are especially important (e.g., therapy applications), collect data with interviews and questionnaires.

User Stories (Section 31.12)

- Write user stories in the form "As a <type of user> I want <some goal> so that <some reason>" to define who, what, and why.
- Create user stories that consist of the INVEST acronym: Independent, Negotiable, Valuable, Estimable, Small, and Testable.

Storyboards (Section 31.13)

- Show users directly interacting in storyboards without worrying about screen layout details since the screen perceptually goes away with fully immersive VR.

Meta Platforms, Inc.
Exhibit 1067
Page 472

- Don't limit storyboards to be linear. Storyboard frames might be connected in multiple ways.

Scope (Section 31.14)

- In addition to stating what will be done, explicitly state what will not be done so that the team can focus on what is important.

Requirements (Section 31.15)

- Requirements convey expectations of the customer; have the client and/or other key players actively involved in defining requirements.
- Requirements should be written in easy-to-understand language that can be understandable by all parties involved.
- Each requirement should be a single thing the application should do.
- Each requirement should be complete, verifiable, and concise, yet with room for innovation and change.
- Any attempt to formulate all possible requirements at the start of a project will fail and would cause considerable delays.
- Work with the client and/or partners when changes are necessary.
- Include universal requirements such as fading the screen out when latency falls above some threshold or when tracking is lost.

The Make Stage (Chapter 32)

- Makers should not develop VR applications without full access to VR hardware.
- Use existing frameworks and tools unless there is a good reason not to do so.

Task Analysis (Section 32.1)

- Task analysis provides organization and structure for description of user activities, which make it easier to describe how activities fit together.
- Use task analysis to understand what tasks must be carried out, to document descriptions of activities, and to look for how to improve processes.
- When the goal is to replicate actions in the real world (e.g., training applications), start with the real-world task instead of the not yet created VR task.
- Find representative users by looking for users who match previously created personas.

Meta Platforms, Inc.
Exhibit 1067
Page 473

- Interview users, domain experts, and visionary representatives to gain insight into what users need and expect.
- Administrate questionnaires, observe experts in action, review existing documentation, and observe VR users trying prototypes.
- Carefully think about each stage of the cycle of interaction.
- Ask "how" questions to break tasks into subtasks.
- Ask "why" questions to get high-level task descriptions and context.
- Ask what happens before and what happens after to obtain sequential information.
- Use graphical depictions when talking with those whom information is being extracted from in order to collect higher quality information.
- Don't be fooled into thinking there is a perfect way to do task analysis.
- After organizing and structuring the task, review with those users that information was solicited from to verify understanding.

Design Specification (Section 32.2)

- In the early stages of specifying the design, explore different designs before converging and committing to a final solution.

Sketches (Section 32.2.1)

- Good sketches are quick and timely, inexpensive and disposable, plentiful, suggestive and explorative, and partially ambiguous. Sketches also have a distinct gestural style, have an appropriate degree of refinement, and contain minimal detail.

Block Diagrams (Section 32.2.2)

- Use block diagrams to show the overall relationship of components without concern for the details.

Use Cases (Section 32.2.3)

- Define use cases to help identify, clarify, and organize interactions in a way that makes them easier to implement.
- Outline use cases with hierarchical levels to provide both high-level and more detailed views. Fill in details as needed.

Meta Platforms, Inc.
Exhibit 1067
Page 474

- Use color and/or different font types to easily see what needs to be implemented, what has already been implemented/tested, differences between paths, and/or who is assigned to what steps.

Classes (Section 32.2.3)

- Organize information with common structure and functionality from earlier steps into themes and then turn those themes into classes.
- Use class diagrams to directly state what developers need to implement.

Software Design Patterns (Section 32.2.5)

- Reuse software design patterns to solve commonly occurring software architecture problems.

System Considerations (Section 32.3)

System Trade-Offs and Making Decisions (Section 32.3.1)

- Don't try to support everything by implementing for the "average" as that will result in no single optimal experience. Instead implement different metaphors and interactions that appropriately fit each option.
- Start by supporting single options and optimize them before adding secondary options.

Support of Different Hardware (Section 32.3.2)

- Attempting to supporting different hardware classes most often results in the experience being optimized for none of them.
- If different hardware classes must be supported, then the core interaction techniques should be independently optimized for each to take advantage of their unique characteristics.

Frame Rate and Latency (Section 32.3.3)

- Maintain a frame rate from the beginning that matches or exceeds the refresh rate of the HMD.
- Carefully observe frame rate as new assets are added and code complexity increases. Even occasional frame drops can be uncomfortable for users.
- Do not only rely on frame rate to determine latency. Measure end-to-end latency using a latency meter.

Meta Platforms, Inc.
Exhibit 1067
Page 475

Sickness Guidelines (Section 32.3.4)

- It is as important for programmers to respect adverse health effects as much as anyone on the team even if they are not prone to sickness themselves.
- Don't wait for the Learn Stage to fix sickness inducing problems.

Calibration (Section 32.3.4)

- Enable easy or automatic calibration—proper calibration is essential.

Simulation (Section 32.4)

Separate Simulation from Rendering (Section 32.4.1)

- Perform rendering asynchronously from simulation.
- Slow simulation updates are often acceptable as long as rendering from head motion occurs with low latency.
- For realistic physics simulations, use a fast update rate.

Fighting Physics (Section 32.4.2)

- The solution to fighting physics is to not fight. Do not attempt to have two separate policies simultaneously determine where an object is.
- Stop applying simulated forces to an object when the object is picked up by the user.

Jittering Objects (Section 32.4.3) and Flying Objects (Section 32.4.4)

- Use physics "cheats" to reduce object jittering and objects flying off into space.
- Avoid having multiple physically-simulated objects interacting whenever possible, especially if the objects are in an enclosed space or are tightly constrained against one another (e.g., a stack of blocks in a small pit or multiple boxes joined together in a circular fashion).

Networked Environments (Section 32.5)

Ideals of Networked Environments (Section 32.5.1)

- Focus on minimizing divergence, causality violations, and expectation violations while maximizing responsiveness and perceived continuity.

Meta Platforms, Inc.
Exhibit 1067
Page 476

Message protocols (Section 32.5.2)

- Use UDP when low latency is a priority, updates occur often, and each update is not essential. Examples of when to use UDP are when continually updating character position and audio.
- Use TCP when state information is only sent once (or occasionally) to ensure the receiving computer can update its state of the world to match the sender's state.

Network Architecture (Section 32.5.3)

- For fast responsiveness, use peer-to-peer networking.
- To ensure all users' worlds are consistent, use a fully authoritative server.
- Use hybrid architectures to take advantage of both peer-to-peer architectures and client-server architectures.

Determinism and Local Estimation (Section 32.5.4)

- For partially deterministic actions, use dead-reckoning to estimate where remotely controlled entities are currently located.
- Use interpolation to reduce perceptual discontinuities where objects appear as jumping to new locations when new network packets are received.

Reducing Network Traffic (Section 32.5.5)

- To reduce network traffic, compute divergence of the local and true state and only send updates when divergence surpasses some threshold. Relevance filtering can also be used to only send relevant information to each computer.
- Conduct audio stress tests by having many users speak simultaneously.
- Use relevance filtering to only send audio to nearby users.
- If necessary, allocate less bandwidth to users who are further away or not in the general forward-looking direction.
- Use animations whenever possible instead of continuous updates.

Simultaneous Interactions (Section 32.5.6)

- Use an object token that can only be owned by a single player at a time to prevent users from simultaneously interacting with the same object.

Meta Platforms, Inc.
Exhibit 1067
Page 477

Networking Physics (Section 32.5.7)

- When simulating physics across networks, have a single authoritative computer own the simulation (although other computers can estimate the simulation).

Prototypes (Section 32.6)

- Use prototypes to learn what users do, not just what they say.
- Begin with minimal prototypes, i.e., start with the least amount of work necessary to receive meaningful feedback.
- Focus on getting something working as quickly as possible and then modify and add additional functionality/features on to the minimal prototype or restart from scratch.
- Have a clear goal for each prototype.
- When building minimal prototypes, give up on trying to look good. Expect that it won't be done well on the first several attempts.

Forms of Prototypes (Section 32.6.1)

- As a first step, consider using real-world prototypes (e.g., physical props and no digital technology) with team members acting out roles.
- Use Wizard of Oz prototypes where an unseen team member enters commands on a workstation.
- Get representational user prototypes working as soon as possible and get feedback from as many naive users as possible.
- Focus on building the prototype that best collects data for what is being targeted.

Final Production (Section 32.7)

- When final production begins, stop exploring possibilities and stop adding more features. Save those ideas for a later deliverable.
- Be open to feedback from stakeholders but make it clear that there is a limit of what is possible within time and budget constraints. There are trade-offs involved and adding new features may come at the cost of delaying other features.

Meta Platforms, Inc.
Exhibit 1067
Page 478

Delivery (Section 32.8)

Demos (Section 32.8.1)

- Before traveling to give a demo, set the demo up in a room different from where equipment is normally located to make sure no equipment is missed when packing.
- Pack two backups of everything!
- Set up in the demo space the day before and confirm everything is working in the morning.
- Be prepared for onsite demos for when important people show up that can add value to the project. This does not mean anyone should be given a demo at any time; demos can take up valuable resources that could be devoted to the project.
- Have a single person be responsible for maintaining and updating onsite demos. Have a single person (possibly the demo maintainer) be responsible for scheduling demos.

Onsite Installation (Section 32.8.2)

- Do not expect installation for non-expert VR customers to go smoothly.
- For complicated installations with multiple pieces of hardware, be prepared to calibrate, modify source code, and use electrical/gaffer tape. Issues might include electromagnetic interference and incompatible connections/cables with a client's system.
- Offer a service to train staff in using and maintaining the system. What may be obvious to you may not be obvious to them.
- Offer support and updates as part of a contract.

Continuous Delivery (Section 32.8.3)

- When delivering online, consider offering builds on a weekly basis. This will force the team to be held accountable and to focus on what is important, as well as to provide plenty of opportunity for data collection.

The Learn Stage (Chapter 33)

- Utilize VR experts, subject-matter experts, usability experts, experiment-design experts, and statisticians to ensure you are doing the right things to maximize learning.

Meta Platforms, Inc.
Exhibit 1067
Page 479

- Lean toward fast feedback, data collection, and experimentation. This enables the team to learn quickly how well or not ideas meet goals, and to correct course immediately.

Communication and Attitude (Section 33.1)

- Seek criticism as if the success of the project and your career depends on it. Because it does!
- Have a positive attitude toward constructive criticism.
- Think positively for yourself, team members, and users.
- Consider failure to be a learning experience and do not fear it.
- Do not blame/belittle users or their opinions/interactions.
- Actively investigate difficulties to determine how the project can improve.
- Assume what others are doing is partially correct and then provide suggestions that enables them to correct and move on.
- When someone points out a problem you are already aware of, thank them for noticing, and encourage them to continue looking for problems.

VR Creators are Unique Users (Section 33.1.1)

- Do not assume what works for you will work for everyone else. You likely won't try all cases and have grown immune to motion sickness.
- Programmers should at least occasionally participate in data collection so that they take feedback more seriously.

Research Concepts (Section 33.2)

Data Collection (Section 33.2.1)

- Do not get stuck in collecting a single type of data. There are many measures that can be used.
- First collect qualitative data to get a high-level understanding before collecting more objective quantitative data.

Reliability (Section 33.2.2)

- Never depend on a single measurement.
- Measure across different participants, sessions, times, and experiments to gain confidence that there is some consistent characteristic for that which is being studied.

Meta Platforms, Inc.
Exhibit 1067
Page 480

- Remember the true score can never be exactly known, although a large number of measurements can provide a good estimate.

Validity (Section 33.2.3)

- Make sure you are measuring and comparing what you think you are measuring and comparing.
- Ideally, a measure should cover all aspects of the construct; correlate with other measures of the same construct; not be affected by variables not related to the construct; and accurately predict other measurement, behavior, and performance.
- A violation of construct validity has occurred when a system artifact has been measured when that is not the intention of the measurement.
- Do not conclude a concept is not effective in general just because it does not work in its current implementation.
- Reduce the chance of coming to incorrect conclusions by carefully considering threats to internal validity. For example, randomize assignment of conditions, keep the setup the same across data collection sessions, remove practice effects, remove any cues that enable participants to guess the hypothesis, and make the experimenter blind to experimental conditions.
- Be aware of threats to statistical conclusion validity to reduce the chances of making erroneous conclusions. For example, a statistical result may have happened by chance, not finding a result does not mean a result does not exist, assumptions of data may not be valid, and data mining increases the odds of finding something by chance.
- Do not assume a finding holds across different settings, other users, other times, and other designs/implementations.
- Collect data again after hardware has changed or significant design changes have occurred.

Sensitivity (Section 32.2.4)

- Increase the sensitivity to that which is being tested in order to increase the chance of finding an effect if it exists.

Constructivist Approaches (Section 33.3)

Mini-Retrospectives (Section 33.3.1)

- Conduct mini-retrospectives—short team discussions of what is going well and what needs improvement.

Meta Platforms, Inc.
Exhibit 1067
Page 481

- It is better to have many short and consistent mini-retrospectives than occasional long retrospectives.
- Respect all team members and never make a retrospective a witch hunt.
- Focus on creating themes for future iterations and areas to track that the team wants to improve upon.

Demos (Section 33.3.2)

- Distinguish the difference between collecting user data (where the primary goal is to learn) and giving demos (where the primary goal is to market). Do not rely solely on demos for receiving feedback.

Interviews (Section 33.3.3)

- An interview results in the best data when given immediately after the person being interviewed experienced the VR application.
- Create interview guidelines in advance in order to keep the interview on track and to reduce bias.
- Attempt to match the interviewer with the intended audience based on previously created personas.
- Conduct interviews in comfortable and natural settings (e.g., a mock living room).
- Don't spend more than 30 minute per interview. Set time limits by scheduling interviews back to back. Individuals can be followed up with if they have more useful things to say.

Questionnaires (Section 33.3.4)

- Use questionnaires when ease of administration and privacy are important.
- Ask for background information to determine if participants fit the target population defined by previously created personas, to help better define personas, to make sure a breadth of users is providing feedback, to match participants when comparing between participants, and to look for correlations of experience with performance.

Focus Groups (Section 33.3.5)

- Use focus groups in the early phases of VR design to explore new concepts, better refine questions, and improve upon data collection processes.
- Group settings are often more efficient than individual interviews and can stimulate thinking as participants build off of each other's ideas.

Meta Platforms, Inc.
Exhibit 1067
Page 482

Expert Evaluations (Section 33.3.6)

- When done properly, expert evaluations are the most efficient and cost-effective method of improving upon the usability of a system and for iterating toward ideal solutions.
- Start early in the project with expert guidelines-based evaluations in order to correct issues before they affect other aspects of the design.
- Conduct formative usability evaluations during the formative and evolving stages of design in order to assess, refine, and improve usability, learning, performance, and exploration.
- Formative usability evaluation should be conducted by VR usability experts as it relies heavily on a solid understanding of VR interaction that vary depending on context (i.e., it consists of more than a predefined list of items to evaluate).
- Pay special attention to critical incidents and make it highest priority to fix issues causing these incidents.
- Design a scoring system apparent to users in order to both increase motivation and to target data collection.
- During the final stages of formative usability evaluation, only observe and do not suggest how to interact with the system.
- Use comparative evaluation to compare two or more well-formed complete or near-complete implementations. This is useful for choosing the best techniques for final integration and to test if a new system is better than a previous system (e.g., Does the new VR training system result in more productivity than traditional training?).

After Action Reviews (Section 33.3.7)

- Debrief users with an emphasis on their specific actions they took during the VR experience.
- Discuss what happened, why it happened, and how it could be done better.
- Discuss with the user the actions she was taking while watching a recording from both first person and third person views.

The Scientific Method (Section 33.4)

- Even if you do not conduct formal experiments, learn the basics in order to design more informal experiments, to comprehend research conducted by others,

Meta Platforms, Inc.
Exhibit 1067
Page 483

to be aware of some of the pitfalls that might occur in collecting data, and to properly interpret results.

- The first step of exploring a problem is to obtain an understanding of what is to be studied. Do this by learning from others (via trying their applications, reading research reports/papers, and talking with them); by trying oneself and observing others interacting with existing VR applications; by constructivist approaches; and by the various concepts discussed in the Define and Make Stages.
- Once familiar with the overall problem, start by asking and answering high-level questions.
- Precisely state the hypothesis in a way that predicts the relationship between two variables.
- Be careful of confounding factors. Consider threats to internal validity to find confounding factors. Once found, set confounds to be constant with control variables.
- Use a within subjects design when only a small number of participants are available and carryover effects (e.g., learning, fatigue, sickness) are minimal and/or participants can be counterbalanced.
- Use a between subjects design when many participants are available, their availability is short, and carryover effects are a concern.
- Always conduct informal pilot studies before conducting a more formal and rigorous experiment in order to help determine feasibility, discover unexpected challenges, improve upon the experiment design, reduce time and cost, and estimate effect size.
- When random assignment is difficult, consider quasi-experiments instead of true experiments at the risk of adding confounding factors.

Data Analysis (Section 33.5)

Making Sense of the Data (Section 33.5.1)

- When first looking at data, start by looking for patterns but be careful of assuming a pattern is the truth as fishing for data can lead to false conclusions.
- Pay attention to outliers as an outlier can provide more insight than more typical data. Often an outlier is an important signal telling you to fix an error.
- The result of a single experiment is rarely the generalized truth. Do not base final decisions on a single experiment and consider how changes may affect results.

Meta Platforms, Inc.
Exhibit 1067
Page 484

Statistical Concepts (Section 33.5.2)

- All team members should be familiar with basic statistical concepts so a common language is spoken, wrong assumptions and errors can be reduced, and research from others can be understood.
- At least one member of the team should be good at statistical analysis.
- Pay attention to measurement types and only perform appropriate calculations on each. For example, do not perform multiplication on interval data and do not add ordinal data.
- For averages, use the median when data is skewed to not take into account outliers or when data is ordinal. Use the mode when data is categorical.
- Visualize data with histograms to intuitively understand the spread of data.
- Understand the differences of statistical significance and practical significance. Statistical significance is not necessarily meaningful.

Meta Platforms, Inc.
Exhibit 1067
Page 485

PART

THE FUTURE STARTS NOW

The ones who are crazy enough to think that they can change the world are the ones who do.

—Steve Jobs

The book has focused on providing a high-level overview of VR and dived into detail on some of the most important concepts. Even for those who have read this book in its entirety, it is only a starting point. There is much more detailed information in the provided references, and even those references are a small proportion of VR research that has occurred over the years. Then there is the general study of neuroscience, human physiology, human-computer interaction, human performance, human factors, etc. The list goes on. However, this in no way means that everything is understood. The field of VR and its many applications is completely wide open. We are in no danger of running out of new possibilities. And hopefully we never run out of those possibilities.

In the words of one of my former professors Henry Fuchs from his IEEE Virtual Reality Keynote, "we now have the opportunity to change the world—let's not blow it!" (Fuchs, 2014). The future is here now and it is accessible by anyone, so don't be left behind. It is up to you to use VR technology in whatever way you wish in order to help shape the future.

Meta Platforms, Inc.
Exhibit 1067
Page 486

Part VII consists of two chapters that look into the future and provide some basic steps for how to get started in actually creating VR applications.

Chapter 35, The Present and Future State of VR, discusses where VR currently stands and where it may lead. Topics include the culture of VR, new generative languages, new indirect and direct ways of interacting, standards and open source, emerging hardware, and the convergence of AR with VR.

Chapter 36, Getting Started, is a short chapter explaining how to get started building VR applications. A schedule of tasks is provided that shows how anyone, even those with limited time, can quickly create new virtual worlds.

Meta Platforms, Inc.
Exhibit 1067
Page 487

The Present and Future State of VR

In 2015, low-cost consumer VR technology has surpassed professional VR/HMD systems. A couple of years ago, even with an unlimited budget, you could not buy a system with the resolution, field of view, low latency, low weight, and overall quality that is now available at a price that anyone can afford. In addition, tools are accessible by anyone and are quite easy to use. This is resulting in the democratization of VR where anyone can take part in defining where VR is going. VR is no longer a tool only for academics or corporate entities. The best VR technologies and experiences are just as likely to be created by a team of indie developers than by a Fortune 500 company.

Some of challenges described in this chapter will open the door for the greatest opportunities over the next few years and beyond. This chapter describes some of the most exciting possibilities where VR is already going and what the future of VR might look like.

35.1 Selling VR to the Masses

Although VR is certainly an exciting industry to be in and many people know about it, the general population is not yet buying it. Instead, VR is being sold to early adopters as a technical marvel. However, technology is not what is important for most people. In the future, the manner in which the story, emotions, and benefits are sold will need to be more compelling to reach a wider audience. How will this be done in a way that is more engaging to the masses so that they demand VR?

- Through entertainment experienced in a way no other technology can provide?
- Through networked worlds that ease and enhance social sharing?
- Through fulfilling people's needs and making life easier?
- Through enhancing quality of life via immersive health care and physical/mental exercise?

Meta Platforms, Inc.
Exhibit 1067
Page 488

- Through cost savings and increased profitability?
- Through new entrepreneurship ventures that we can't yet even begin to fathom?

Focusing on any one of these is certainly a noble effort. Each will likely play a big part in changing many people's lives.

35.2 Culture of the VR Community

In some ways, VR culture is already forming. Consider the VR Meetup phenomena.[1] In 2012, Karl Krantz, a self-proclaimed introvert, began to contact what he thought would be a small number of local VR professionals and enthusiasts to start a VR Meetup to share and discuss their creations. Today, Karl has evolved into somewhat of a VR celebrity. Not only has his local Meetup expanded to over 2000 members, but his model of allowing anyone to showcase their VR experiences at public events has been replicated throughout the world. Pockets of hundreds of VR enthusiasts in every major city gather to share their virtual worlds and discuss the latest VR trends on a regular basis. Non-VR Meetups often struggle to have more than a few people attend their events, yet all one needs to do for a VR Meetup is to set up an account, find a location, set a date, send out some tweets and emails, and people will flock to the event. In the San Francisco Bay Area alone, there are so many different VR Meetups per week that it is difficult to keep up with all of them. Whereas the popular media has portrayed VR as being anti-social, these Meetups have proven VR to be the social lubricant that gets normally introverted individuals passionately conversing.

Both Brendan Iribe, CEO of Oculus, and Mark Zuckerburg, CEO of Facebook, claim there will someday be a billion or more VR users (Zuckerberg, 2014; Hollister, 2014). In order for this to happen, an entire cultural shift is going to be required for the masses to accept such advanced mind altering technology. Will this actually happen? It is certain to happen at some point as both humans and technology evolve. The question is will that be 5 years from now or 50 years from now? In the short term, will it only be the fortunate few who care as is the case in Neal Stephenson's Snow Crash or will VR scale more quickly? Even if the Metaverse is not accessed by the hundreds of millions, various forms of VR will certainly add value to some smaller groups of us.

Regardless of who uses VR, cultural patterns of rules, hierarchies, and social constraints will certainly form within virtual worlds. Some of these will be coded into the experiences but much of the social and cultural structure will evolve naturally just as they do in the real world. What actions will be considered rude and what actions will

1. To find a local VR Meetup, go to http://www.meetup.com.

Meta Platforms, Inc.
Exhibit 1067
Page 489

be considered polite? What will the penalty be for unacceptable behavior? Will the equivalent of government entities form that impose laws for the common good? Only time will tell, but these questions have important implications for the future of VR and should be considered when designing new worlds.

35.3 Communication

The book began with a discussion of how VR at its core is communication (Section 1.2). This section describes how communication in VR will be taken to the next level beyond just interacting with objects that is done with most current VR applications.

35.3.1 A New Generative Language

Languages are not necessarily made of words, but are any form of communication (e.g., body language). A **Generative language** (also called future-based language), in contrast to a descriptive language, has "the power to create new futures, to craft vision, and to eliminate the blinders that are preventing people from seeing new possibilities" (Zaffron and Logan, 2009). In short, generative language transforms the world. VR can be thought of as a new generative language because we are creating new experiences that words cannot fully describe and can only be fully conveyed by actually experiencing it oneself. The language of VR will certainly bring about things that would never be created from traditional language alone, our past experiences, or our current understanding.

35.3.2 Symbolic Communication

Direct communication is not always the best way to interact within VR. Indirect symbolic communication is the use of abstract symbols (e.g., text) to represent objects, ideas, concepts, quantities, etc. (Bowman et al., 2004). Consider how we use indirect symbolic communication in the real world. Symbolic communication enables us to convey information efficiently, precisely, and concisely; provides a way to clearly think, both inside our minds and in the physical world (e.g., paper and whiteboards); and allows structured data to persist throughout time.

In VR, symbolic output is quite easy to do and is used quite effectively. However, symbolic input is not so straightforward. Other than the widgets and panels interaction techniques described in Section 28.4.1 that work well for small numbers of options, symbolic input for more generalized input is rarely used. There are currently no obvious ways to symbolically input data in a way that is effective, efficient, and elegant.

Meta Platforms, Inc.
Exhibit 1067
Page 490

Inventing new widgets can certainly help. Consider a touch wheel interface on a 2D tablet touch screen for entering dates. Such a widget works extremely well for touch-screens but is not great for a mouse/keyboard interface. We need equivalent symbolic input techniques specifically for VR. And we need to go beyond that. Theoretically, speech, gestures, chord keyboards, and pinch input are ideal for symbolic input. However, there are multiple practical challenges to getting such theoretical concepts to work well within VR.

Speech recognition works well in controlled conditions. However, as many can attest to, talking to a voice prompt system via telephone can be extremely frustrating. Even if a system perfectly recognized the spoken word, there still remain significant challenges. In addition to semantic parsing and understanding, challenges include the context words are spoken in (consider the phrase "Put it to the left"), some people being resistant to using such systems due to lack of privacy, a perception of bothering others, and feeling awkward talking with a machine. It remains to be seen if speech recognition becomes more common within VR.

Gestures have similar challenges as voice recognition. In addition to error rates, fatigue can be an issue. Better tracking of the entire hand will certainly help so large dynamic gestures will not be required as is the case with Microsoft Kinect. It remains to be seen if reliability from camera-based systems is inherently a largely unsolvable physics problem due to line of sight challenges or if that can somehow be worked around. Gloves may become more common as gesture recognition improves and users accept the inherent encumbrance of wearing hardware. Attempts have been made with sign language recognition but have failed due to inaccurate and unreliable finger tracking. However, as gesture recognition continues to improve, sign language may indeed become a useful method of symbolic input. A place to start could be with numeric input since only a small number of gestures would need to be recognized and easy to learn since we already understand how to count with our fingers. The system must recognize more gestures than just the digits 0–9, such as to start numeric entry and delete an entry, but not many.

Many challenges have nothing to do with technical capability. The best design does not always win. Consider how smart phones first took input from a 12-digit matrix phone design that maps multiple presses on each digit to the alphanumeric input (or word completion that attempts to guess what the user intended by mapping multiple presses to words in a dictionary). Today, smartphones still take symbolic input through a QWERTY touch interface that was originally designed for 10 fingers, but is now used with one or two fingers. Swipe pattern recognition as the finger stays on the screen makes it more efficient but this interface is certainly not ideal. Like the smart phone industry, transition to better symbolic input within VR will take time and will evolve along a non-optimal path. Such a slow response to designing for the

Meta Platforms, Inc.
Exhibit 1067
Page 491

medium has already been proven to also be the case for VR, as most VR widgets are just replications of desktop widgets.

Touch can be extremely important for symbolic input. Haptics will help by conveying feedback such as pushing a key on a virtual keyboard. Buttons on chord keyboards inherently provide physical feedback from the feel of the button. Such capability can easily be integrated into hand-held tracked controllers. Pinch gloves also provide the sense of touch when two fingers touch. However, there is a steep learning curve to learning complex new input patterns. If there becomes enough value and demand for reliable symbolic input then perhaps users will accept the pain of change (such as people were willing to learn to use typewriters).

35.3.3 Creating Empathy with Virtual Humans

Science fiction has predicted humans would live and interact with computer entities that resemble humans in natural ways, even to the point of developing an emotional relationship with those artificial entities. That is now already becoming a reality. Researchers from the University of Southern California Institute for Creative Technology have developed such human-like entities complete with interactive social skills and connection/rapport-building capability. The system works by sensing the user's emotional state and responding appropriately for psychotherapy applications (Figure 35.1). Those who believed they were interacting with a computer-controlled virtual human versus a human-controlled virtual human reported a lower fear of self-disclosure, reported lower impression management (only disclosing positive information), displayed sadness more intensely, and were rated by observers as more willing to disclose (Lucas et al., 2014). Although challenges certainly exist to integrate such a system within fully immersive VR (e.g., how do the sensors detect emotions conveyed on the face when half the face is covered by an HMD?), it is certainly only a matter of time when such capability is fully integrated into fully immersive VR. With fully immersive VR, virtual humans appear more life size and real, rather than being confined to a screen, interacting with users as if they were real entities as envisioned by science fiction writers. It is unknown where this might eventually lead, but there is certainly potential to integrate artificial intelligent entities with VR that could result in helping real people overcome barriers that occur with traditional human-to-human communication.

35.3.4 Brain to Brain Communication

Researchers from Neuroelectrics and other institutions recently conducted a proof-of-concept experiment where individuals were able to directly convey information through brain-to-brain communication without intervention of motor or peripheral sensory systems (Figure 35.2) (Grau et al., 2014). Binary streams of encoded words were

Meta Platforms, Inc.
Exhibit 1067
Page 492

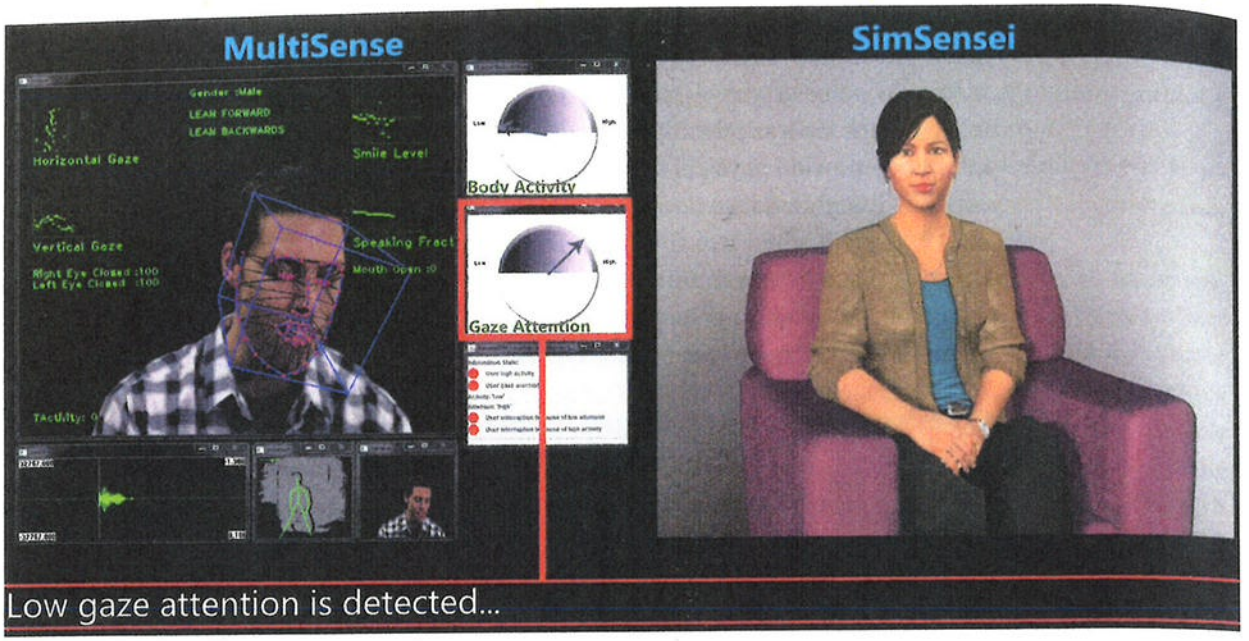


Figure 35.1 **A computer-controlled virtual human (right) builds empathy with a user (left) by sensing his state and responding appropriately.** (Courtesy of USC Institute for Creative Technologies. Principal Investigators: Albert (Skip) Rizzo and Louis-Philippe Morency)

transmitted between the remote minds of emitter and receiver participants, representing the realization of a human brain-to-brain interface. This was done by capturing voluntary motor imagery-controlled electroencephalographic (EEG) changes from one participant, encoding the information, and sending it from India to a remote participant in France through the Internet. The information was transformed to signals that conveyed to the receiving participant the conscious perception of phosphenes (light flashes) through neuronavigated, robotized transcranial magnetic stimulation (TMS).

The results provided a demonstration for the development of conscious brain-to-brain communication technologies. More fully developed implementations will open new research venues in cognitive, social, and clinical neuroscience and the scientific study of consciousness. Long-term possibilities include being able to modify one's sensory VR experience by thought alone. Such communication technologies will certainly have a profound impact on more than just VR technology.

Meta Platforms, Inc.
Exhibit 1067
Page 493

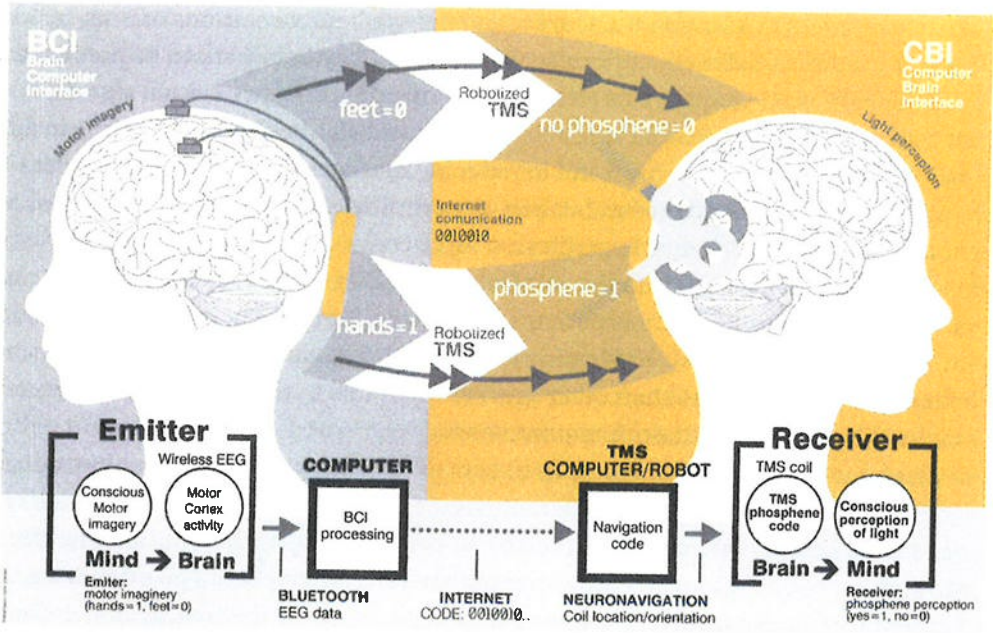


Figure 35.2 The brain-to-brain communication system (Grau et al., 2014).

35.3.5 Full Neural Input and Output

Direct neural input through some standardized connection, as done in the Matrix Triology, is certainly many years away. The transition will be very gradual, starting with neural input and output as done with Neuroelectrics described above.

Visual displays are already moving closer to the brain via virtual retinal display where a laser draws images onto the retina (Schowengerdt, 2003) as done by the company Magic Leap. Visual displays will eventually move to contact lenses, starting with augmented reality, but eventually providing the capability to not only display but to completely block out the real world. The next step will be either a projector located inside the eye or via signals sent directly to the retina. In fact, multiple companies have already started to pursue direct retinal stimulation. Simple retinal stimulators providing a matrix display of 60 "pixels" have already been successfully implanted in blind patients, enabling them to see basic shapes, motions, and even letters/words in the real world (via mapping from a digital camera mounted on glasses to the implanted electrodes) (da Cruz, 2013).

Things become exceedingly complex beyond the retina as the parvo and magno cells transmitting signals to the superior colliculus and lateral geniculate nucleus

Meta Platforms, Inc.
Exhibit 1067
Page 494

(Section 8.1.1) are not simply 2D representations of images. Ideas of how to transmit directly to the visual cortex is far beyond what can be predicted today, but it will likely require completely rethinking the concept of visual signals.

Virtual vestibular input would also be quite challenging to do in an accurate/precise way to be useful in reducing motion sickness. Artificial input to the vestibular system exists today, but in a very primitive form that adds to sickness and balance more than it helps. Regardless, if more control can be provided to the vestibular system, or more directly to the vestibular nuclei, then this may someday be feasible.

There is perhaps less reason for direct audio input as headphones can already be largely hidden from view inside the ear. However, if there is a need this may be easier to do than other sensory input due to the more simplified nature of audio and lower bandwidth requirements as compared to vision.

Neural output to control objects within the virtual world and/or the real world is an entirely different story. There is already demand for such capability for those who are disabled. Prosthetic devices are already controlled by electromyography signals. One can easily imagine such technology could be extended to augmented and virtual reality. In fact, one only need to attend the Neurogaming Conference and Expo held every year in San Francisco to control a basic VR interface via thought alone.

35.4 Standards and Open Source

Some consider standards to be controversial. At one extreme there are those that say any and all standards hinder innovation whereas there are others that want to standardize everything. In truth, standards are simply a tool that, if used wisely, can simplify life for everyone, ease communication across groups, foster development across multiple platforms, and enable consumers to more objectively compare between competing products. Standards can also provide a catalyst for industry leaders to regularly get together, collaborate, and upgrade their findings as they progress—which can enhance innovation rather than hinder it. If standards blocked innovation, the automotive industry, the gaming industry, the Internet, and even VR would not be where it is now.

When defining standards, it's a misnomer that they only refer to interoperability, which is often the main issue of contention within industry. Interoperability is only part of the puzzle as there are other fundamental standards that need to be pursued. Standards can help with quality expectations, basic language, health and safety, and even defining common goals.

Precise and consistent terminology is one area where standards would be useful. For example, most people casually discuss field of view without specifying if that is the

Meta Platforms, Inc.
Exhibit 1067
Page 495

horizontal field of view or the diagonal field of view (among other factors). Similarly, latency is not well defined. Contrary to what many people believe, latency is more than simply the inverse of the frame rate or refresh rate (Section 15.4). Furthermore, in a raster display, is latency from the time of some action until the time that the resulting first pixel in the upper-left corner in a frame (the start of the raster scan) is displayed or until the time the frame's center pixel is displayed (a difference of 8 ms for a 60 Hz display)? Or, what about the response time of the pixel? Depending on the technology used, pixel response can take several milliseconds, if not tens of milliseconds, to completely reach 100% of its intended intensity. Even if pixel response is immediate, what if the pixel persists for some time? Is latency up to the time the pixel initially appears or up to the average time that it is visible? Such detailed differences can make a big difference when discussing latency. Add these differences due to imprecise/unstandardized definitions together and the variability can easily be 20 ms.

These are some of the most basic elements of VR, and yet the industry players are far apart from one another, and this contributes to customer misunderstanding and unmet expectations. The importance of standards is just as much about interoperability between people and ideas as they are about compatibility and commonalities between vendors.

35.4.1 Open Source

Open Source is a class of licensing that usually pertains to software. It works by allowing everyone to contribute to a given project that is free to use and transparent for all to see (Schneider, 2015). When something is published as open source, it immediately gives others the right to use the same code from that point on, though they can't charge money for it. The idea can be rewritten from scratch and sold outside of the open source world, but modifications of existing code have to be transparent and free.

Open Source Interoperability

In the 1990s, Russ Taylor led the development of the VR Peripheral Network (VRPN), an open source device-independent and network-transparent system for communicating with VR devices (Taylor et al., 2001a). VRPN has since become the most often-used software library for connecting VR devices to VR applications. Russ claims one reason for its success is that "you can only standardize what nobody cares about" (Taylor et al., 2001b). VR creators do care about having a standard way of connecting with VR devices where different devices can be used with the same application, but they do not care about the low-level detail of how that is accomplished.

Meta Platforms, Inc.
Exhibit 1067
Page 496

OSVR (Open Source VR) is a collaboration that is owned and maintained by Razer and Sensics. It's not a formal organization or non-profit structure, and is instead a platform that is defined by signed licensing agreements between Razer and the participating vendors. OSVR is promoted as having approximately 150 commercial hardware developers, game studios, and research institutions involved.

Russ is now one of the primary developers for Open Source VR (OSVR) platform that includes both open source software and hardware designs so that anyone can freely build their own or modify existing systems.

35.4.2 Platform-Specific/De Facto Standards

A de facto standard is a system or platform that has achieved a dominant position due to the sheer volume of people associated to the product that is achieved by public acceptance or market forces. For example, the Microsoft XBOX and PC use the DirectX de facto standard. Similarly, Valve has Steam OS. Advanced Micro Devices (AMD) have their own LiquidVR platform while Nvidia has GameWorks VR. They are platforms because their standards work with a range of compliant hardware, but they are not open standards.

35.4.3 Open Standards

Open standards are "standards made available to the general public and are developed (or approved) and maintained via a collaborative and consensus driven process" (ITU-T, 2015).

What is key is collaboration and agreement across a wide range of individuals and all participants have equal voting rights regardless of size. Contributions can't be thrown in and taken out unless there is adequate group discussion and agreement. The Khronos Group has open source platforms, but the finalized spec is determined by the membership through votes. Without balance and equal voting share, huge innovations can get missed because the larger participants don't want to collaborate or they simply outgun the smaller players.

Unless all VR work moves toward a platform which a single vendor controls, the most effective open standards are backed by a formal non-profit organization, and everyone must be welcome to participate with equal rights. Unless this happens, the effort will fall apart and lose its credibility. Non-profit structures exist because they force developments to be open and transparent (Mason, 2015).

Notable Open Standard Organizations

The Khronos Group is a good working example of a non-profit standards organization, and their work is free of royalty fees and licensing fees. OpenGL, OpenCL, WebGL, and

Meta Platforms, Inc.
Exhibit 1067
Page 497

more are all credited to The Khronos Group. To participate, members pay an annual fee, decisions are voted on, and the standards are implemented.

The Immersive Technology Alliance (http://www.ita3d.com) is a formal non-profit corporation that was originally founded in 2009 under a different name. Its Executive Director is Neil Schneider who also created Meant to be Seen (mtbs3D.com). MTBS is where the Oculus Rift was born and marks the spot where John Carmack and Palmer Luckey first met. The ITA exists to make immersive technology successful, and in addition to having its own working groups pertaining to standards and industry growth, it regularly collaborates with external organizations like SIGGRAPH, The Khronos Group, Meetup organizations (e.g., SVVR), and more. They have also launched Immersed Access (http://www.immersedaccess.com), an NDA-backed private community for professionals to share and learn from one another in a safe environment that is closed off from press. Immersed Access features unofficial discussion areas for OSVR, Oculus, Valve, and other platforms.

35.5 Hardware

New VR hardware developments are now occuring on a regular basis largely due to access of 3D Printing. HMDs are becoming lighter and with wider fields of view. Hybrid tracking is improving precision and accuracy for both the head and the hands. Magic Leap claims to be solving the accommodation-vergence conflict. Numerous companies are improving upon full body tracking. Eventually, exoskeletons may be built to provide better haptics and to complement human abilities with superhuman powers.

The leading VR companies Oculus, Valve, Sony, and Sixense are all now creating tracked hand-held controller options at low price points (at least compared to older professional systems). Such controllers are currently the best way to interact with a majority of fully immersive VR experiences (although such applications are still relatively rare due to the lack of user ownership of such devices). This is significant as not having hands in VR is equivalent to being paralyzed in the real world. As these types of hand input devices become more available, then developers will create better and more innovative methods of interacting. At some point most all VR experiences, other than completely passive experiences, will enable users to interact with their hands.

Even with 6DoF hand input devices, current hand input has been described as a limiting "boxing glove" style interface where fingers are rarely tracked, and if fingers are tracked, they are rarely tracked accurately and never with 100% reliability. Tyndall is working in collaboration with TSSG and NextGen Interactions on commercializing their extremely accurate glove technology, previously used for surgical training. The

Meta Platforms, Inc.
Exhibit 1067
Page 498

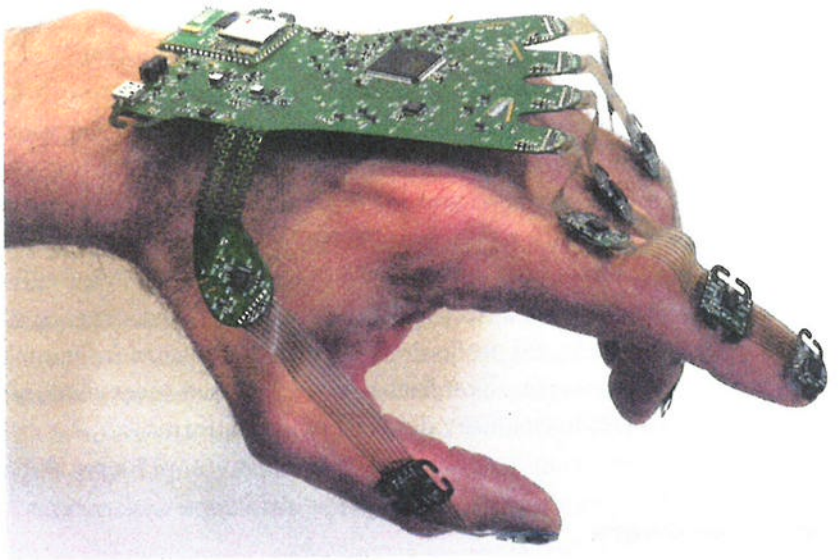


Figure 35.3 **Early prototype of the Tyndall/TSSG VR Glove.**

goal is to provide a VR glove at a consumer price-point that overcomes the significant challenges previous gloves have faced. Figure 35.3 shows a prototype of the glove.

35.6 The Convergence of AR and VR

Augmented reality and virtual reality have many differences, yet there are also many similarities and the two will likely converge. Although the experiences may still be very different (VR is transporting someone to a different world whereas augmented reality adds to the local world), the same hardware might be used for both. For example, cameras have been capturing the real world and bringing it into the non-see-through HMD VR experience since the 1990s (known as augmented virtuality; see Section 3.1). So the convergence of AR and VR is not new from a research perspective, but would be new as far as catching mainstream attention. Conversely, future optical-see-through HMDs will be able to make individual pixels opaque so digital imagery can completely occlude all or part of the real world.

Meta Platforms, Inc.
Exhibit 1067
Page 499

Getting Started

VR technology is moving at lightning speed. It was only in 2011 when most people outside of academics, corporate research, and science fiction communities thought VR was an obsolete joke. Fast forward to today where thousands of independent developers, startups, and Fortune 500 Companies are creating VR experiences, positioning themselves as VR leaders. The message is clear: If a company wants to be competitive there is no time to waste on writing the perfect project plan lest they risk being left behind. It's a new world, one where the prototype should have been completed last week and informal feedback from an initial demo should have happened yesterday. Yes, spend some time on the Define Stage of the project. But spend a couple of days versus a month on the initial plan before building something. Or better yet jump straight to the Make Stage. Having no software development experience is no excuse—basic prototypes can be built with today's tools by pointing and clicking with a mouse. The content need not be novel—this is just for you to get started. Experiment by looking around, then modify and experiment with a few things. Then move back to the Define Stage by sketching some ideas out on paper. After all, if a teenager with no budget can build a basic VR experience that his friends and family are impressed by, then an adult committed to changing the future can do the same. Anyone that can't put together a basic plan, spend a few hundred dollars on hardware, create a prototype experience (or better yet, do so by working with a colleague or friend), and demonstrate that prototype to some acquaintances for feedback is not yet serious about positioning themselves as a leader of the VR movement. For those of who are ready, **welcome to the new reality!**

Here is a very reasonable schedule for someone who can build VR experiences *part time in little as a few hours per week.* If you have somehow managed to arrange your life or convinced your boss to do this full time, then the limiting factor will be the arrival of the hardware. If you have experience in software development then you can easily condense this schedule to under a week. So stop making excuses and **get to work!**

Meta Platforms, Inc.
Exhibit 1067
Page 500

- Week 1
  - Find and attend a local VR Meetup (http://www.meetup.com). Try some demos and talk to as many people as possible. Get to know those that are also serious about developing VR experiences. Collect their contact information.
  - Order an HMD.
- Week 2
  - While waiting for the hardware to arrive, download Unity (it is free for individuals or companies making under $100K/year) or your development tool of choice.
  - Search online for some non VR tutorials and start working through them in order to become comfortable with the core tools (even if you don't plan on being a developer, you should know the basics).
- Week 3
  - Once the HMD arrives, follow the directions that came with the HMD. Try the manufacturer's basic demos.
  - Find and download some different VR experiences. Note what you like and don't like.
- Week 4
  - Using Unity or your tool of choice, add a texture mapped plane, a cube floating in space, and a light source (I have performed the initial Define Stage for you!).
  - Build the application, put on the HMD, and look around.
  - Take off the HMD and modify the scene. Build and put on the HMD noticing how things have changed.
  - Congratulations, you have performed the most basic form of the define-make-experiment iteration in perhaps a couple of hours. It doesn't matter how bad it is at this point, you are learning fast.
  - Now iterate some more!
- Week 5
  - Write a basic concept for your first project based on what you liked and didn't like about the experiences you tried and your first scene that you built. Be creative.
  - Don't worry about details such as requirements, constraints, and task analysis yet. That can come later.

Meta Platforms, Inc.
Exhibit 1067
Page 501

- Week 6
  - Build some low-level functionality for your basic concept. Don't worry about the art yet. Remember this is a prototype to test your ideas.
  - Show to friends, family, and people you met at the Meetup. Get their feedback of what works and what doesn't work.
- Week 7 and beyond
  - Iterate. Iterate. Iterate. For each iteration, expand upon each stage or start completely over at any time.

Congratulations—you are now a VR creator and a contributor to the VR revolution! This is not the end, for it is only the beginning.

Meta Platforms, Inc.
Exhibit 1067
Page 502

Example Questionnaire

Below is a questionnaire from NextGen Interactions used for evaluation of MakeVR (formerly Studio-In Motion) in collaboration with Digital ArtForms and Sixense (Jerald et al., 2013). The questionnaire was distributed to users after they constructed a castle within the application that took approximately two hours per user. The questionnaire includes a Kennedy Sickness Questionnaire (Section 16.1), a Lickert Scale (Section 33.3.4), previous experience, and an open-ended questionnaire inquiring about what worked and what could be improved.

Meta Platforms, Inc.
Exhibit 1067
Page 503

Studio In-Motion Questionnaire Page **1 of 4**

Simulator Sickness Questionnaire

Do you feel that you are in the same state good health as when you started the experiment? If you answered no please explain briefly in the space provided below: ☐ Yes ☐ No

For each of the following conditions, please circle how you are feeling right now, on the scale of *none* through *severe*.

1. General discomfort	*none*	*slight*	*moderate*	*severe*
2. Fatigue (weariness or exhaustion of the body)	*none*	*slight*	*moderate*	*severe*
3. Headache	*none*	*slight*	*moderate*	*severe*
4. Eye strain (weariness or soreness of the eyes)	*none*	*slight*	*moderate*	*severe*
5. Difficulty focusing	*none*	*slight*	*moderate*	*severe*
6. Increased salivation	*none*	*slight*	*moderate*	*severe*
7. Sweating	*none*	*slight*	*moderate*	*severe*
8. Nausea (stomach distress)	*none*	*slight*	*moderate*	*severe*
9. Difficulty concentrating	*none*	*slight*	*moderate*	*severe*
10. Fullness of head (sinus pressure)	*none*	*slight*	*moderate*	*severe*
11. Blurred vision	*none*	*slight*	*moderate*	*severe*
12. Dizzy (with eyes open)	*none*	*slight*	*moderate*	*severe*
13. Dizzy (with eyes closed)	*none*	*slight*	*moderate*	*severe*
14. Vertigo (surroundings seem to swirl)	*none*	*slight*	*moderate*	*severe*
15. Stomach awareness (just a short feeling of nausea)	*none*	*slight*	*moderate*	*severe*
16. Burping	*none*	*slight*	*moderate*	*severe*

If you expressed slight, moderate, or severe on any of the questions above, please state if you felt that way before using the system and if so, explain how you felt worse after using the system.

You should not drive an automobile for at least one half-hour after the end of the experiment.

Meta Platforms, Inc.
Exhibit 1067
Page 504

Studio In-Motion Questionnaire **Page 2 of 4**

Likert Scale

Mark a single box (*strongly disagree, disagree, undecided, agree,* or *strongly agree*) for each of questions below.

	strongly disagree	*disagree*	*undecided*	*agree*	*strongly agree*
I found the interface easy to learn.	☐	☐	☐	☐	☐
Once I learned the interface, I found the navigation to be easy and intuitive to use.	☐	☐	☐	☐	☐
Once I learned the interface, I found object manipulation easy and intuitive to use.	☐	☐	☐	☐	☐
Once I learned the interface, I found that I could focus on being creative instead of on the technicalities of the interface.	☐	☐	☐	☐	☐
I prefer the new interface to a mouse and keyboard interface.	☐	☐	☐	☐	☐
I was able to create the objects/scenes that I wanted.	☐	☐	☐	☐	☐
I would use the system myself in my home.	☐	☐	☐	☐	☐
I would recommend the system to friends.	☐	☐	☐	☐	☐
Overall, I enjoyed the experience.	☐	☐	☐	☐	☐

Demographic Information

1. Age and gender

   Age _____

   Gender _____

Meta Platforms, Inc.
Exhibit 1067
Page 505

Studio In-Motion Questionnaire **Page 3 of 4**

Background/Experience

For each question, please put a check next to your answer.

1. How often do you use a computer in your average week?

   I use computers less than . . .

   ☐ 1. 1 hour ☐ 5. 20 hours
   ☐ 2. 2 hours ☐ 6. 40 hours
   ☐ 3. 5 hours ☐ 7. more than 40 hours
   ☐ 4. 10 hours

2. Over the past two years, what is the most you have played video games in a single week? Select one.

   I played less than . . .

   ☐ 1. 1 hour ☐ 5. 20 hours
   ☐ 2. 2 hours ☐ 6. 40 hours
   ☐ 3. 5 hours ☐ 7. more than 40 hours
   ☐ 4. 10 hours

3. How many times have you used systems where yon did not use your hands in free 3D space to control the computer application (e.g., not using a keyboard, mouse, or touchscreen).

   Before today I used free 3D space interfaces . . .

   ☐ 1. Never ☐ 5. 11–20 times
   ☐ 2. 1 time ☐ 6. 20–100 times
   ☐ 3. 2 times ☐ 7. nearly every day at school, at work, or for entertainment
   ☐ 4. 5–10 times

4. How much experience do you have with CAD (Computer Aided Design) software (e.g., Maya, 3D Studio, Sketchup)

   Before today I used CAD . . .

   ☐ 1. Never ☐ 5. 11–20 times
   ☐ 2. 1 time ☐ 6. 20–100 times
   ☐ 3. 2 times ☐ 7. nearly every day at school, at work, or for entertainment
   ☐ 4. 5–10 times

Meta Platforms, Inc.
Exhibit 1067
Page 506

Studio In-Motion Questionnaire **Page 4 of 4**

Open-Ended Questions

1. What did you most like about the system?

2. How long did it take you to get good at using the system?

3. Were you able to create what you wanted to create?

4. Could you have created a more interesting scene given more time?

5. What did you dislike about the system?

6. What suggestions or ideas do you have for improving the system?

7. Did you feel tired from using the systems? If so, in what areas of your body did you feel fatigue?

8. What do do you think consumers would pay for this system?

9. Any other comments?

Meta Platforms, Inc.
Exhibit 1067
Page 507

Example Interview Guidelines

Interview Question Guidelines

NextGen Interactions
Consulting & Contracting Services

Interview prep. Turn on the video camera and mic.
Test to make sure video and sound is recording properly.

Written questionnaire. Expand on the answers provided in the exit questionnaire.

General comments. Open-ended questions of free thought and discussion.

Health. How do they feel? If not well, is there any insight when it started to occur or what caused it?

Presentation. Any comments on the audio, visual, or other cues? Did anything not look, sound, or feel good?

Engagement and presence. Did they feel engaged in the virtual world? Did they feel like they were looking at/observing the experience or did they feel like they were a part of the experience? Were there any specific problems or events that broke the illusion?

Understanding and intuitiveness. Was the experience understandable? Was it obvious what they were expected to do? What specifically was difficult to understand?

Time usage. Did they want to use the system more? Or was the session too long or too tiring?

Ease of use and difficulties. Were the tasks easy or difficult to interact with? Any comments on the difficulty of using the system? What specifically could be improved upon?

Missing features and ideas. What would they have liked to experience that was not available? What would be some useful things to include in a future version?

Other comment and suggestions. Any other comments or suggestions to make the system and experience better?

Thank them for participating. Ask them if they would like to participate in future events and/or provide further feedback. Collect contact information.

Meta Platforms, Inc.
Exhibit 1067
Page 509

Glossary

[to be supplied]

References

Abrash, M. (2013). Why Virtual Reality is Hard (and where it might be going). In *Game Developers Conference*. Retrieved from http://media.steampowered.com/apps/valve/2013/MAbrashGDC2013.pdf

Adelstein, B. D., Burns, E. M., Ellis, S. R., and Hill, M. I. (2005). Latency Discrimination Mechanisms in Virtual Environments: Velocity and Displacement Error Factors. In *Proceedings of the 49th Annual Meeting of the Human Factors and Ergonomics Society* (pp. 2221–2225).

Adelstein, B. D., Lee, T. G., and Ellis, S. R. (2003). Head Tracking Latency in Virtual Environments: Pyschophysics and a Model. In *Proceedings of the 47th Annual Meeting of the Human Factors and Ergonomics Society* (pp. 2083–2987).

Adelstein, B. D., Li, L., Jerald, J., and Ellis, S. R. (2006). Suppression of Head-Referenced Image Motion During Head Movement. In *Proceedings of the 50th Annual Meeting of Human Factors and Ergonomics Society* (pp. 2678–2682).

Alonso-Nanclares, L., Gonzalez-Soriano, J., Rodriguez, J. R., and DeFelipe, J. (2008). Gender differences in human cortical synaptic density. *Proceedings of the National Academy of Sciences USA, 105*, 14615–14619. DOI: 10.1073/pnas.0803652105

Anstis, S. (1986). Motion Perception in the Frontal Plane: Sensory Aspects. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbook of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Anthony, S. (2013, April 2). Kinect-based system diagnoses depression with 90% accuracy. *ExtremeTech.com*. Retrieved from http://www.extremetech.com/extreme/152309-kinect-based-system-diagnoses-depression-with-90-accuracy

Arditi, A. (1986). Binocular Vision. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbook of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Attneave, F., and Olson, R. K. (1971). Pitch as a medium: a new approach to psychophysical scaling. *The American Journal of Psychology, 84*, 147–166.

Meta Platforms, Inc.
Exhibit 1067
Page 511

item Azuma, R. T. (1997). A Survey of Augmented Reality. *PRESENCE: Teleoperators and Virtual Environments*, *6*(4), 355–385.

Azuma, R. T., and Bishop, G. (1995). A Frequency-Domain Analysis of Head-Motion Prediction. In *Proceedings of ACM SIGGRAPH 95* (pp. 401–408). ACM Press.

. Vision and Virtual Badcock, D. R., Palmisano, S., and May, J. G. (2014). Vision and Virtual Environments. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed.). Boca Raton, FL: Boca Raton, FL: CRC Press.

Bailey, R. (1995, March 8). Costs are Dropping for Virtual Reality Systems. *New York Times*. Retrieved from http://www.nytimes.com/1995/03/08/news/08iht-ff.html

Balakrishnan, R., and Hinckley, K. (2000). Symmetric bimanual interaction. *Human Factors in Computing Systems*, *2*(1), 33–40. DOI: 10.1145/332040.332404

Bhalla, M., and Proffitt, D. R. (1999). Visual-motor recalibration in geographical slant perception. *Journal of Experimental Psychology. Human Perception and Performance*, *25*, 1076–1096. DOI: 10.1037/0096-1523.25.4.1076

Human Bliss, J. P., Proaps, A. B., and Chancey, E. T. (2014). Human Performance Measurements in Virtual Environments. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 749–780). Boca Raton, FL: CRC Press.

Bolas, M., Kuruvilla, A., Chintalapud, S., Rabelo, F., Lympouridis, V., Suma, E., Barron, C. Matamoros, C., Debevec, P. (2015). Creating Near-Field VR Using Stop Motion Characters and a Touch of Light-Field Rendering. In *ACM SIGGRAPH Posters*.

Bolas, M. T. (1989). *Design and Virtual Environments*. Stanford.

Bolas, M. T. (1992). Design and Virtual Environments. In *International Conference on Artificial Reality and Tele-Existence* (pp. 135–141).

Bolt, R. A. (1980). "Put-that-there": Voice and gesture at the graphics interface. In *SIGGRAPH* (pp. 262–270). DOI: 10.1145/800250.807503

Botvinick, M., and Cohen, J. (1998). Rubber hands "feel" touch that eyes see. *Nature*, *391*(6669), 756. DOI: 10.1038/35784

Bowman, D. A., and Hodges, L. (1997). An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. In *ACM Symposium on Interactive 3D Graphics2* (pp. 35–38). ACM Press.

Bowman, D. A., Kruijff, E., Jr., J. J. L., and Poupyrev, I. (2004). *3D User Interfaces: Theory and Practice*. Addison Wesley. Retrieved from http://books.google.com/books?id=JYzmCkf7yNcC&pgis=1

Bowman, D. A., McMahan, R. P., and Ragan, E. D. (2012). Questioning naturalism in 3D user interfaces. *Communications of the ACM*, *55*(9), 78–88. DOI: 10.1145/2330667.2330687

Meta Platforms, Inc.
Exhibit 1067
Page 512

Bowman, D. A., and Wingrave, C. A. (2001). Design and evaluation of menu systems for immersive virtual environments. In *IEEE Virtual Reality* (pp. 149–156). DOI: 10.1109/VR.2001.913781

Bowman, D. (1999). Interaction Techniques for Common Tasks in Immersive Virtual Environments. *Techniques*. DOI: 10.1.1.29.6639

Bowman, D., Davis, E., Badre, A., and Hodges, L. Maintaining Spatial Orientation during Travel in an Immersive Virtual Environment. *Presence: Teleoperators and Virtual Environments,* vol. 8, no. 6, 1999, pp. 618-631.

Bowman, D., Koller, D., and Hodges, L. A Methodology for the Evaluation of Travel Techniques for Immersive Virtual Environments. *Virtual Reality: Research, Development, and Applications,* vol. 3, no. 2, 1998, pp. 120–131.

Brewster, D. (1856). *The stereoscope: its history, theory, and construction, with its Application to the fine and useful arts and to education.* London: John Murray.

Bridgeman, B., Heijden, A. H. C. Van der, and Velichkovsky, B. M. (1994). A Theory of Visual Stability Across Saccadic Eye Movements. *Behavioral and Brain Sciences*, *17*(2), 247–292.

Brooks, F. (1988). Grasping Reality Through Illusion: Interactive Graphics Serving Science. In *CHI'88* (pp. 1–11). DOI: 10.1145/57167.57168

Brooks, F. P. (2010). *The Design of Design: Essays From a Computer Scientist*. Addison-Wesley.

Brooks, F. P., Ouh-Young, M., Batter, J. J., and Jerome Kilpatrick, P. (1990). Project GROPE Haptic displays for scientific visualization. *ACM SIGGRAPH Computer Graphics*, *24*(4), 177–185. DOI: 10.1145/97880.97899

Brooks, J. O., Goodenough, R. R., Crisler, M. C., Klein, N. D., Alley, R. L., Koon, B. L., Logan, W. C., Ogle, J. H., Tyrrell, R. A., Wills, R. F. (2010). Simulator sickness during driving simulation studies. *Accident Analysis and Prevention*, *42*, 788–796. DOI: 10.1016/j.aap.2009.04.013

Brooks Jr., F. P. (1995). *The Mythical Man Month: Essays on Software Engineering* (2nd ed.). Addison-Wesley.

Bryson, S., and Johan, S. (1996). Time Management, Simultaneity and Time-Critical Computation in Interactive Unsteady Visualization Environments. In *Proceedings of IEEE Visualization '96* (pp. 255–262). IEEE Computer Science Press.

Burns, E., Razzaque, S., Panter, A. T., Whitton, M. C., McCallus, M. R., and Brooks, F. P., Jr. (2006). The hand is more easily fooled than the eye: users are more sensitive to visual interpenetration than to visual-proprioceptive discrepancy. *Presence: Teleoperators and Virtual Environments - Special Issue: IEEE VR 2005*, *15*(1), 1–15.

Buxton, B. (2007). *Sketching User Experiences: getting the design right and the right design. Sketching User Experiences*. Focal Press. DOI: 10.1016/B978-012374037-3/50064-X

Meta Platforms, Inc.
Exhibit 1067
Page 513

Buxton, W. (1986). There's more to interaction than meets the eye: Some issues in manual input. In D. A. Norman and S. W. Draper (Eds.), *User Centered System Design: New perspectives on human-computer interaction* (pp. 319–337). Lawrence Erlbaum Associates. Retrieved from http://www.billbuxton.com/eye.html

Empirical Comparison of Pie vs. Linear Menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 95–100). DOI: 10.1145/57167.57182

Carmack, J. (2015). 2015 GDC - Carmack on Mobile VR (Full Livestream Recording). Retrieved April 20, 2015; from https://www.youtube.com/watch?v=UNAmAxT7-qs

Chance, S. S., Gaunet, F., Beall, A. C., and Loomis, J. M. (1998). Locomotion Mode Affects the Updating of Objects Encountered During Travel: The Contribution of Vestibular and Proprioceptive Inputs to Path Integration. *Presence: Teleoperators and Virtual Environments*. DOI: 10.1162/105474698565659

Chen, S. E. (1995). Quicktime {VR}—An Image-based Approach to Virtual Environment Navigation. In *Proceedings of ACM SIGGRAPH 95* (pp. 29–38). ACM Press.

Clulow, F. W. (1972). *Color: Its principle and their applications.* New York: Morgan and Morgan.

Cohn, M. (2005). *Agile Estimating and Planning*. Prentice Hall.

Comeau, C. P., and Brian, J. S. (1961, November). Headsight television system provides remote surveillance. *Electronics*, 86–90.

Coren, S., Ward, L. M., and Enns, J. T. (1999). *Sensation and Perception* (5th ed.). Harcourt Brace College Publishers.

Costello, P. (1997). *Health and Safety Issues Associated with Virtual Reality: A Review of Current Literature*. JISC Advisory Group on Computer Graphics (AGOCG) Technical Report Series No. 37, Computing Services, Loughborough University.

Craig, A. B., Sherman, W. R., and Will, J. D. (2009). *Developing Virtual Reality Applications. Developing Virtual Reality Applications*. DOI: 10.1016/B978-0-12-374943-7.00005-7

Cruz-Neira, C., Sandin, D. J., DeFanti, T. A., Kenyon, R. V., and Hart, J. C. (1992). The CAVE: audio visual experience automatic virtual environment. *Communications of the ACM*. DOI: 10.1145/129888.129892

Crystal, A., and Ellington, B. (2004). Task analysis and human-computer interaction?: approaches, techniques, and levels of analysis. In *America's Conference on Information Systems* (pp. 1–9). Retrieved from http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1967&context=amcis2004

Csikszentmihalyi, M. (2008). *Flow: The psychology of optimal performance. Optimal experience: Psychological studies of flow in consciousness*. Harper Perennial Modern Classics.

Meta Platforms, Inc.
Exhibit 1067
Page 514

Cunningham, D. W., Billock, V. A., and Tsou, B. H. (2001a). Sensorimotor Adaptation to Violations in Temporal Contiguity. *Psychological Science*, *12*(6), 532–535.

Cunningham, D. W., Chatziastros, A., Heyde, M. Von Der, and Bulthoff, H. H. (2001b). Driving in the Future: Temporal Visuomotor Adaptation and Generalization. *Journal of Vision*, *1*(2), 88–98.

Cutting, J. E., and Vishton, P. M. (1995). Perceiving Layout and Knowing Distances?: The Integration, Relative Potency, and Contextual Use of Different Information about Depth. In W. Epstein and S. Rogers (Eds.), *Perception of Space and Motion* (Vol. 22, pp. 69–117). San Diego, CA: Academic Press. DOI: 10.1016/B978-012240530-3/50005-5

da Cruz, L., Coley, B. F., Dorn, J., Merlini, F., Filley, E., Christopher, P., et al (2013). The Argus II epiretinal prosthesis system allows letter and word reading and long-term function in patients with profound vision loss. *The British journal of ophthalmology*, 97(5), 632–636.

Daily, M., Howard, M., Jerald, J., Lee, C., Martin, K., McInnes, D., and Tinker, P. (2000). Distributed design review in virtual environments. In *Proceedings of the third international conference on Collaborative virtual environments* (pp. 57–63). ACM. Retrieved from DOI: 10.1145/351006.351013

Daily, M., Sarfaty, R., Jerald, J., McInnes, D., and Tinker, P. (1999). The CABANA: A Reconfigurable Spatially Immersive Display. In *Projection Technology Workshop* (pp. 123–132).

Dale, E. (1969). *Audio-Visual Methods in Teaching* (3rd ed.). The Dryden Press.

Darken, R., Cockayne, W., and Carmein, D. (1997). The Omni-directional Treadmill: A Locomotion Device for Virtual Worlds. In *ACM Symposium on User Interface Software and Technology* (pp. 213–221). DOI: 10.1145/263407.263550

Darken, R. P. (1994). Hands-off interaction with menus in virtual spaces. In *SPIE Stereoscopic Displays and Virtual Reality Systems* (pp. 365–371). DOI: 10.1117/12.173893

Darken, R. P., and Cevik, H. (1999). Map usage in virtual environments: orientation issues. *Proceedings IEEE Virtual Reality (Cat. No. 99CB36316)*. DOI: 10.1109/VR.1999.756944

Darken, R. P., and Peterson, B. (2014). Spatial orientation, wayfinding, and representation. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 467–491). Boca Raton, FL: CRC Press.

Darken, R. P., and Sibert, J. L. (1993). A toolset for navigation in virtual environments. *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology UIST93*, *2740*, 157–165. DOI: 10.1145/168642.168658

Darken, R. P., and Sibert, J. L. (1996). Wayfinding strategies and behaviors in large virtual worlds. *CHI '96*, 142–149. DOI: 10.1145/238386.238459

Meta Platforms, Inc.
Exhibit 1067
Page 515

Davis, S., Nesbitt, K., and Nalivaiko, E. (2014). A Systematic Review of Cybersickness. In *Proceedings of the 2014 Conference on Interactive Entertainment* (pp. 1–9). Newcastle, NSW, Australia: ACM.

Debevec, P., Downing, G., M., B., Peng, H., and Urbach, J. (2015). Spherical Light Field Environment Capture for Virtual Reality using a Motorized Pan/Tilt Head and Offset Camera. In *ACM SIGGRAPH Posters*.

Delaney, D., Ward, T., and McLoone, S. (2006a). On Consistency and Network Latency in Distributed Interactive Applications: A Survey—Part I. *Presence: Teleoperators and Virtual Environments*, *15*(2), 218–234. DOI: 10.1162/pres.2006.15.2.218

Delaney, D., Ward, T., and McLoone, S. (2006b). On Consistency and Network Latency in Distributed Interactive Applications: A Survey—Part II. *Presence: Teleoperators and Virtual Environments*, *15*(4), 465–482.

DiZio, P., Lackner, J. R., and Champney, R. K. (2014). Proprioceptive Adaptation and Aftereffects. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 835–856). Boca Raton, FL: CRC Press.

Dodgson, N. A. (2004). Variation and extrema of human interpupillary distance. In *SPIE Stereoscopic Displays and Virtual Reality Systems 5291* (pp. 36–46).

Dorst, K., and Cross, N. (2001). Creativity in the design process: Co-evolution of problem-solution. *Design Studies*, *22*(5), 425–437. DOI: 10.1016/S0142-694X(01)00009-6

Drachman, D. A. (2005). Do we have brain to spare? *Neurology*, *64*, 2004–2005. DOI: 10.1212/01.WNL.0000166914.38327.BB

Draper, M. H. (1998). *The Adaptive Effects of Virtual Interfaces: Vestibulo-Ocular Reflex and Simulator Sickness*. University of Washington.

Duh, H. B., Parker, D. E., and Furness, T. A. I. (2001). An 'Independent Visual Background' Reduced Balance Disturbance Evoked by Visual Scene Motion: Implication for Alleviating Simulator Sickness. In *Proceedings of ACM CHI 2001* (pp. 85–89).

Eastgate, R. M., Wilson, J. R., and D'Cruz, M. (2014). Structured Development of Virtual Environments. In Kelly S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 353–389). Boca Raton, FL: CRC Press.

Ebenholtz, S. M., Cohen, M. M., and Linder, B. J. (1994). The possible role of nystagmus in motion sickness: A hypothesis. *Aviation Space and Environmental Medicine*, *65*(11), 1032–1035.

Eg, R., and Behne, D. M. (2013). Temporal integration for live conversational speech. In *Auditory-Visual Speech Processing* (pp. 129–134).

Ellis, S. R. (2014). Where are all the Head Mounted Displays? Retrieved April 14, 2015, from http://humansystems.arc.nasa.gov/groups/acd/projects/hmd_dev.php

Meta Platforms, Inc.
Exhibit 1067
Page 516

Ellis, S. R., Mania, K., Adelstein, B. D., and Hill, M. I. (2004). Generalizeability of Latency Detection in a Variety of Virtual Environments. In *Proceedings of the 48th Annual Meeting of the Human Factors and Ergonomics Society* (pp. 2083–2087).

Ellis, S. R., Young, M. J., Adelstein, B. D., and Ehrlich, S. M. (1999). Discrimination of Changes in Latency During Head Movement. In *Proceedings of Human-Computer Interaction 99* (pp. 1129–1133). L. Erlbaum Associates, Inc.

Fann, C.-W., Jiang, J.-R., and Wu, J.-W. (2011). Peer-to-Peer Immersive Voice Communication for Massively Multiplayer Online Games. *2011 IEEE 17th International Conference on Parallel and Distributed Systems*, 759–764. DOI: 10.1109/ICPADS.2011.99

Ferguson, E. S. (1994). *Engineering and the Mind's Eye*. The MIT Press.

Foley, J., Dam, A., Feiner, S., and Hughes, J. (1995). *Computer Graphics: Principles and Practice*. Reading, MA: Addison-Wesley Publishing.

Fuchs, H. (2014). Telepresence: Soon not just a dream and a promise. In *IEEE Virtual Reality*.

Gabbard, J. L. (1997). *A taxonomy of usability characteristics in virtual environments*. Virginia Tech.

Gabbard, J. L. (2014). Usability Engineering of Virtual Environments. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 721–747). Boca Raton, FL: CRC Press.

Ganel, T., Tanzer, M., and Goodale, M. A. (2008). A double dissociation between action and perception in the context of visual illusions. *Psychological Science, 19*, 221–225. DOI: 10.1111/j.1467-9280.2008.02071.x

Gautier, L., Diot, C., and Kurose, J. (1999). End-to-end transmission control mechanisms for multiparty interactive applications on the Internet. In *IEEE INFOCOM* (pp. 1470–1479). DOI: 10.1109/INFCOM.1999.752168

Gebhardt, S., Pick, S., Leithold, F., Hentschel, B., and Kuhlen, T. (2013). Extended pie menus for immersive virtual environments. *IEEE Transactions on Visualization and Computer Graphics, 19*(4), 644–51. DOI: 10.1109/TVCG.2013.31

Gibson, J. J. (1933). Adaptation, after-effect and contrast in the perception of curved lines. *Journal of Experimental Psychology*. DOI: 10.1037/h0074626

Gliner, J. A., Morgan, G. A., and Leech, N. L. (2009). *Research Methods in Applied Settings: An Integrated Approach to Design and Analysis* (2nd ed.). Routledge.

Gogel, W. C. (1990). A Theory of Phenomenal Geometry and its Applications. *Perception and Psychophysics, 48*(2), 105–123.

Goldstein, E. B. (2007). *Sensation and Perception* (7th ed.). Belmont, CA: Wadsworth Publishing.

Goldstein, E. B. (Ed.). (2010). *Encyclopedia of Perception*. SAGE Publications.

Meta Platforms, Inc.
Exhibit 1067
Page 517

Goldstein, E. B. (2014). *Sensation and Perception* (9th ed.). Cengage Learning.

Gortler, S. J., Grzeszczuk, R., Szeliski, R., and Cohen, M. F. (1996). The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96* (pp. 43–54). DOI: 10.1145/237170.237200

Gothelf, J., and Seiden, J. (2013). *Lean UX:* (E. Ries, Ed.). O'Reilly Media, Inc.

Grau, C., Ginhoux, R., Riera, A., Nguyen, T. L., Chauvat, H., Berg, M., Amengual, J. L., Pascual-Leone, A., Ruffini, G. (2014). Conscious Brain-to-Brain Communication in Humans Using Non-Invasive Technologies. *PloS One*, *9*(8), e105225. DOI: 10.1371/journal.pone.0105225

Greene, N. (1986). Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications*, *6*(11), 21–29.

Gregory, R. L. (1973). *Eye and Brain: The Psychology of Seeing* (2nd ed.). London: Weidenfeld and Nicolson.

Gregory, R. L. (1997). *Eye and Brain: The Psychology of Seeing* (5th ed.). Princeton University Press.

Guadagno, R. E., Blascovich, J., Bailenson, J. N., and Mccall, C. (2007). Virtual Humans and Persuasion: The Effects of Agency and Behavioral Realism. *Media Psychology*, *10*(1), 1–22. DOI: 10.108/15213260701300865

Guiard, Y. (1987). Asymmetric division of labor in human skilled bimanual action: the kinematic chain as a model. *Journal of Motor Behavior*, *19*(4), 486–517. DOI: 10.1080/00222895.1987.10735426

Haan, G. De, Koutek, M., and Post, F. H. (2005). IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection. In *In Virtual Environments 2005* (pp. 201–209). DOI: 10.1.1.134.2264

Hallett, P. E. (1986). Eye Movements. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbook of Perception and Human Performance* (Vol. 1). New York: Wiley-Interactive.

Hamit, F. (1993). *Virtual Reality and the Exploration of Cyberspace*. Sams Publishing.

Hannema, D. (2001). *Interactions in Virtual Reality*. University of Amsterdam.

Harm, D. L. (2002). Motion Sickness Neurophysiology, Physiological Correlates, and Treatment. In K. M. Stanney (Ed.), *Handbook of Virtual Environments* (pp. 637–661). Mahwah, N. J.: Lawrence Erlbaum Associates.

Harmon, L. (1973). The Recognition of Faces. *Scientific American, 229*(5), 70–84.

Heider, F., and Simmel, M. (1944). An experimental study of apparent behavior. *The American Journal of Psychology*, *57*, 243–259. DOI: 10.2307/1416950

Heilig, M. (1960). Stereoscopic-television apparatus for individual use. Patent US2955156

Meta Platforms, Inc.
Exhibit 1067
Page 518

Heilig, M. L. (1992). El Cine Del Futuro: The Cinema of the Future. *Presence: Teleoperators and Virtual Environments*, *1*(3), 279–294.

Hettinger, L. J., Schmidt-Daly, T. N., Jones, D. L., and Keshavarz, B. (2014). Illusory self-motion in virtual environments. In K. Hale and K. Stanney (Eds.), *Handbook of virtual environments* (2nd ed., pp. 435–465). Boca Raton, FL: CRC Press.

Hinckley, K., Pausch, R., Goble, J. C., and Kassell, N. F. (1994). Passive Real-world Interface Props for Neurosurgical Visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94* (Vol. 30, pp. 452–458). DOI: 10.1145/191666.191821

Hinckley, K., Pausch, R., Proffitt, D., and Kassell, N. F. (1998). Two-handed virtual manipulation. In *ACM Transactions on Computer-Human Interaction* (Vol. 5, pp. 260–302). DOI: 10.1145/292834.292849

Ho, C. C., and MacDorman, K. F. (2010). Revisiting the uncanny valley theory: Developing and validating an alternative to the Godspeed indices. *Computers in Human Behavior*, *26*(6), 1508–1518. DOI: 10.1016/j.chb.2010.05.015

Hoffman, H. G. (2004). Virtual-reality therapy. *Scientific American*, *291*, 58–65. DOI: 10.1038/scientificamerican0804-58

Hollister, S. (2014). Oculus wants to build a billion-person MMO with Facebook. Retrieved April 8, 2015, from http://www.theverge.com/2014/5/5/5684236/oculus-wants-to-build-a-billion-person-mmo-with-facebook

Holloway, R. (1997). Registration Error Analysis for Augmented Reality. *Presence: Teleoperators and Virtual Environments*, *6*(4), 413–432.

Holloway, R. L. (1995). *Registration Errors in Augmented Reality Systems*. Department of Computer Science, University of North Carolina at Chapel Hill.

Homan, R. (1996). SmartScene: An Immersive, Realtime, Assembly, Verification and Training Application. In *Workshop on Computational Tools and Facilities for the Next-Generation Analysis and Design Environment* (pp. 119–140). NASA Conference Publication. Retrieved from https://archive.org/stream/nasa_techdoc_19970014680/19970014680_djvu.txt

Hopkins, A. A. (2013). *Magic: Stage Illusions and Scientific Diversions, Including Trick Photography*. Dover Publications.

Houben, M., and J.Bos. (2010). Reduced seasickness by an artificial 3d earth-fixed visual reference. In *International Conference on Human Performance at Sea* (pp. 263–270).

Howard, I. P. (1986a). The Perception of Posture, Self Motion, and the Visual Vertical. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbook of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Meta Platforms, Inc.
Exhibit 1067
Page 519

Howard, I. P. (1986b). The Vestibular System. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbook of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Howard, M., Tinker, P., Martin, K., Lee, C., Daily, M., and Clausner, T. (1998). The Human Integrating Virtual Environment. In *All-Raytheon Software Symposium.*

Hu, H., Ren, Y., Xu, X., Huang, L., and Hu, H. (2014). Reducing view inconsistency by predicting avatars' motion in multi-server distributed virtual environments. *Journal of Network and Computer Applications*, *40*(1), 21–30. DOI: 10.1016/j.jnca.2013.08.011

Hummels, C., and Stappers, P. J. (1998). Meaningful gestures for human computer interaction: Beyond hand postures. In *Proceedings - 3rd IEEE International Conference on Automatic Face and Gesture Recognition, FG 1998* (pp. 591–596). DOI: 10.1109/AFGR.1998.671012

Hutchins, E., Hollan, J., and Norman, D. (1986). Direct Manipulation Interfaces. In D. A. Norman and S. W. Draper (Eds.), *User-Centered System Design: New Perspectives in Human-Machine Interaction* (pp. 87–124). Boca Raton, FL: CRC Press. DOI: 10.1207/s15327051hci0104_2

Ingram, R., and Benford, S. (1995). Legibility enhancement for information visualisation. In *Proceedings Visualization '95* (pp. 209–216). DOI: 10.1109/VISUAL.1995.480814

Insko, B. E. (2001). *Passive haptics significantly enhances virtual environments.* Ph.D. Dissertation, Department of Computer Science UNC-Chapel Hill.

International Society for Presence Research. (2000). The Concept of Presence: Explication Statement. Retrieved November 6, 2014, from http://ispr.info/

ITU-T. (2015). Definition of "Open Standards." Retrieved June 20, 2015; from http://www.itu.int/en/ITU-T/ipr/Pages/open.aspx

Iwata, H. (1999). Walking about virtual environments on an infinite floor. In *IEEE Virtual Reality* (pp. 286–293). DOI: 10.1109/VR.1999.756964

Jacob, R. J. K. (1991). The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Transactions on Information Systems.* DOI: 10.1145/123078.128728

James, T., and Woodsmall, W. (1988). *Time Line Therapy and the Basis of Personality.* Meta Publications.

Jerald, J. (2009). *Scene-Motion- and Latency-Perception Thresholds for Head-Mounted Displays.* Department of Computer Science, University of North Carolina at Chapel Hill.

Jerald, J., Daily, M. J., Neely, H. E., and Tinker, P. (2001). Interacting with 2D Applications in Immersive Environments. In *EUROIMAGE International Conference on Augmented Virtual Environments and 3d Imaging* (pp. 267–270).

Meta Platforms, Inc.
Exhibit 1067
Page 520

Jerald, J., Fuller, A. M., Lastra, A., Whitton, M., Kohli, L., and Brooks, F. (2007). Latency compensation by horizontal scanline selection for head-mounted displays. *Proceedings of SPIE, 6490*, 64901Q–64901Q–11. Retrieved from http://link.aip.org/link/PSISDG/v6490/i1/p64901Q/s1&Agg=doi

Jerald, J., Marks, R., Laviola, J., Rubin, A., Murphy, B., Steury, K., and Hirsch, E. (2012). The Battle for Motion-Controlled Gaming and Beyond (Panel). In *ACM SIGGRAPH*.

Jerald, J., Mlyniec, P., Yoganandan, A., Rubin, A., Paullus, D., and Solotko, S. (2013). MakeVR: A 3D world-building interface. In *Symposium on 3D User Interfaces* (pp. 197–198).

Jerald, J., Peck, T. C., Steinicke, F., and Whitton, M. C. (2008). Sensitivity to scene motion for phases of head yaws. In *symposium on Applied perception in graphics and visualization* (pp. 155–162). ACM Press. Retrieved from http://portal.acm.org/citation.cfm?doid=1394281.1394310 DOI: 1394281.1394310

Johansson, G. (1976). Spatio-temporal differentiation and integration in visual motion perception. *Psychological Research, 38*(4), 379–393. DOI: 10.1007/BF00309043

Johnson, D. M. (2005). *Simulator Sickness Research Summary*. Fort Rucker, AL: U.S. Army Research Institute for the Behavioral and Social Science.

Jones, J. A., Suma, E. A., Krum, D. M., and Bolas, M. (2012). Comparability of Narrow and Wide Field-Of-View Head-Mounted Displays for Medium-Field Distance Judgments. In *ACM Symposium on Applied Perception* (p. 119).

Kabbash, P., Buxton, W., and Sellen, A. (1994). Two-handed input in a compound task. In *SIGCHI conference on Human factors in computing systems* (pp. 417–423). DOI: 10.1145/191666.191808

Kant, I. (1781). *Critique of Pure Reason. Critique of Pure Reason* (Vol. 2). Retrieved from http://www.gutenberg.org/files/4280/4280-h/4280-h.htm

Kennedy, R. S., and Fowlkes, J. E. (1992). Simulator Sickness Is Polygenic and polysymptomatic: Implications for Research. *The International Journal of Aviation Psychology*. DOI: 10.1207/s15327108ijap0201_2

Kennedy, R. S., Kennedy, R. C., Kennedy, K. E., Wasula, C., and Bartlett, K. M. (2014). Virtual Environments and Product Liability. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 505–518). Boca Raton, FL: CRC Press.

Kennedy, R. S., Lane, N. E., Berbaum, K. S., and Lilienthal, M. G. (1993). A Simulator Sickness Questionnaire (SSQ): A New Method for Quantifying Simulator Sickness. *International Journal of Aviation Psychology, 3*(3), 203–220.

Kennedy, R. S., and Lilienthal, M. G. (1995). Implications of Balance Disturbances Following Exposure to Virtual Reality Systems. In *Proceedings of the IEEE Virtual Reality Annual International Symposium (VRAIS)* (pp. 35–39). IEEE Computer Society Press.

Meta Platforms, Inc.
Exhibit 1067
Page 521

Keshavarz, B., Hecht, H., and Lawson, B. D. (2014). Visually Induced Motion Sickness. In K. Hale and K. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 647–698). Boca Raton, FL: CRC Press.

Kim, Y. Y., Kim, H. J., Kim, E. N., Ko, H. D., and Kim, H. T. (2005). Characteristic changes in the physiological components of cybersickness. *Psychophysiology*, *42*(5), 616–625. DOI: 10.1007/s00234-005-1388-2

Klumpp, R. G. (1956). Some Measurements of Interaural Time Difference Thresholds. *The Journal of the Acoustical Society of America*. DOI: 10.1121/1.1908493

Kohli, L. (2013). *Redirected Touching*. University of North Carolina at Chapel Hill.

Kolasinski, E. (1995). *Simulator Sickness in Virtual Environments*. United States Army Research Institute for the Behavioral and Social Sciences, Technical Report 1027.

Kopper, R., Bowman, D. A., Silva, M. G., and McMahan, R. P. (2010). A human motor behavior model for distal pointing tasks. *International Journal of Human Computer Studies*, *68*(10), 603–615. DOI: 10.1016/j.ijhcs.2010.05.001

Korzybski, A. (1933). *Science and Sanity - an introduction to non-aristotelean systems and general semantics*. European Society for General Semantics.

Krum, D. M., Suma, E. A., and Bolas, M. (2012). Augmented reality using personal projection and retroreflection. *Personal and Ubiquitous Computing*, *16*(1), 17–26. DOI: 10.1007/s00779-011-0374-4

Kumar, M. (2007). *Gaze-enhanced user interface design*. Stanford. Ph.D. Dissertation.

Kurtenbach, G., Sellen, A., and Buxton, W. (1993). An Empirical Evaluation of Some Articulatory and Cognitive Aspects of Marking Menus. *Human-Computer Interaction*. DOI: 10.1207/s15327051hci0801_1

Lackner, J. (2014). Motion sickness: More than nausea and vomiting. *Experimental Brain Research*, *232*, 2493–2510. DOI: 10.1007/s00221-014-4008-8

Lackner, J. R., and Teixeira, R. (1977). Visual-Vestibular Interaction: Vestibular Stimulation Suppresses the Visual Induction of Illusory Self-rotation. *Aviation, Space and Environmental Medicine*, *48*, 248–253.

Lang, B. (2015, January 5). Visionary VR is Reinventing Filmmaking's Most Fundamental Concept to Tell Stories in Virtual Reality. *Road to VR*.

Larman, C. (2004). *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley.

Laviola, J. (1999). *Whole-Hand and Speech Input in Virtual Environments*. Providence, RI: Brown University Press.

LaViola, J. J. (2000). A discussion of cybersickness in virtual environments. *ACM SIGCHI Bulletin*. DOI: 10.1145/333329.333344

Meta Platforms, Inc.
Exhibit 1067
Page 522

LaViola, J., and Zeleznik, R. (1999). Flex and Pinch: A Case Study of Whole-Hand Input Design for Virtual Environment Interaction. In *International Conference on Computer Graphics and Imaging'99* (pp. 221–225).

Lawson, B. D. (2014). Motion Sickness Symptomatology and Origins. In K. Hale and K. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 531–600). Boca Raton, FL: CRC Press.

Lehar, S. (2007). The Constructive Aspect of Visual Perception: A Gestalt Field Theory Principle of Visual Reification Suggests a Phase Conjugate Mirror Principle of Perceptual Computation. Retrieved from http://cns-alumni.bu.edu/~slehar/ConstructiveAspect/ConstructiveAspect.pdf

Leigh, J., Johnson, A., and Defanti, T. A. (1997). CAVERN: A distributed architecture for supporting scalable persistence and interoperability in collaborative virtual environments. *Virtual Reality: Research, Development and Applications*, *2*(2), 217–237.

Levoy, M., and Hanrahan, P. (1996). Light field rendering. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '96*, 31–42. DOI: 10.1145/237170.237199

Li, L., Adelstein, B. D., and Ellis, S. R. (2006). Perception of Image Motion During Head Movement. In *Proceedings of the ACM Symposium on Applied Perception in Graphics and Visualization* (pp. 45–50). Boston, MA: ACM Press.

Liang, J., and Green, M. (1994). JDCAD: A highly interactive 3D modeling system. *Computers and graphics*, 18(4), 499–506.

Lin, J. J. W., Abi-Rached, H., and Lahav, M. (2004). Virtual guiding avatar: An effective procedure to reduce simulator sickness in virtual environments. In *Conference on Human Factors in Computing Systems - Proceedings* (pp. 719–726). Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-4544250250&partnerID=40&md5=e8702da495f6ad516a11210168b1efd0

Lindeman, R., and Beckhaus, S. (2009). Crafting memorable VR experiences using experiential fidelity. *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, 187–190. DOI: 10.1145/1643928.1643970

Lindeman, R. W. (1999). *Bimanual interaction, passive-haptic feedback, 3D widget representation, and simulated surface constraints for interaction in immersive virtual environments*. George Washington University.

Lindeman, R. W., Sibert, J. L., and Hahn, J. K. (1999). Hand-held windows: towards effective 2D interaction in immersive virtual environments. In *IEEE Virtual Reality* (pp. 205–212). DOI: 10.1109/VR.1999.756952

Link. (2015). Link Foundation Advanced Simulation and Training Fellowships. Retrieved April 15, 2015, from http://linksim.org/

Meta Platforms, Inc.
Exhibit 1067
Page 523

Lisberger, S. G., and Movshon, J. A. A. (1999). Visual Motion Analysis for Pursuit Eye Movements in Area MT of Macaque Monkeys. *The Journal of Neuroscience*, *19*(6), 2224–2246.

Loomis, J. M., and Knapp, J. M. (2003). Visual Perception of Egocentric Distance in Real and Virtual Environments. In L. J. Hettinger and M. W. Haas (Eds.), *Virtual and adaptive environments*. Mahwah, NJ: Lawrence Erlbaum Associates.

Loose, R., and Probst, T. (2001). Velocity Not Acceleration of Self-Motion Mediates Vestibular-Visual Interaction. *Perception*, *30*(4), 511–518.

Lucas, G. M., Gratch, J., King, A., and Morency, L. P. (2014). It's only a computer: Virtual humans increase willingness to disclose. *Computers in Human Behavior*, *37*, 94–100. DOI: 10.1016/j.chb.2014.04.043

Mack, A. (1986). Perceptual Aspects of Motion in the Frontal Plane. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbook of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Mack, A., and Herman, E. (1972). A New Illusion: The Underestimation of Distance During Pursuit Eye Movements. *Perception and Psychyphysics*, *12*, 471–473.

Maltz, M. (1960). *Psycho-Cybernetics*. Prentice-Hall.

Mania, K., Adelstein, B. D., Ellis, S. R., and Hill, M. I. (2004). Perceptual Sensitivity to Head Tracking in Virtual Environments with Varying Degrees of Scene Complexity. In *Proceedings of the ACM Symposium on Applied Perception in Graphics and Visualization* (pp. 39–47).

Mapes, D., and Moshell, J. (1995). A Two-Handed Interface for Object Manipulation in Virtual Environments. *Presence: Teleoperators and Virtual Environments*, *4*(4), 403–416.

Mark, W. R., McMillan, L., and Bishop, G. (1997). Post-rendering 3D warping. In *Proceedings of Interactive 3D Graphics* (pp. 7–16). Providence, RI.

Marois, R., and Ivanoff, J. (2005). Capacity limits of information processing in the brain. *Trends in Cognitive Sciences*. DOI: 10.1016/j.tics.2005.04.010

Martin, J. (1998). TYCOON: Theoretical Framework and Software Tools for Multimodal Interfaces. In J. Lee (Ed.), *Intelligence and Multimodality in Multimedia Interfaces*. AAAI Press.

Mason, W. 2015, Open Source vs. Open Standard, an important distinction, Retrieved July 2, 2015 from http://uploadvr.com/osvr-may-be-open-source-but-it-is-not-open-standard-and-that-is-an-important-distinction-says-neil-schneider-of-the-ita/

May, J. G., and Badcock, D. R. (2002). Vision and Virtual Environments. In K. M. Stanney (Ed.), *Handbook of Virtual Environments* (pp. 29–63). Mahwah, NJ: Lawrence Erlbaum Associates.

Meta Platforms, Inc.
Exhibit 1067
Page 524

McCauley, M. E., and Sharkey, T. J. (1992). Cybersickness: Perception of Motion in Virtual Environments. *Presence: Teleoperators and Virtual Environments, 1(3)*, 311–318. Retrieved from http://md1.csa.com/partners/viewrecord.php?requester=gs&collection=TRD&recid=A9349402AH&q=http://www.csa.com/partners/viewrecord.php?requester=gs&collection=TRD&recid=A9349402AH&uid=791564131&setcookie=yes

McGurk, H., and MacDonald, J. (1976). Hearing lips and seeing voices. *Nature, 264*, 746–748.

McMahan, R. P. (2007). *Exploring and Evaluating Task Sequences for System Control Interfaces in Immersive Virtual Environments*. Virginia Polytechnic Institute. Retrieved from http://scholar.lib.vt.edu/theses/available/etd-06132007-143300/

McMahan, R. P., and Bowman, D. A. (2007). An empirical comparison of task sequences for immersive virtual environments. In *IEEE Symposium on 3D User Interfaces* (pp. 25–32). DOI: 10.1109/3DUI.2007.340770

McMahan, R. P., Bowman, D. A., Zielinski, D. J., and Brady, R. B. (2012). Evaluating display fidelity and interaction fidelity in a virtual reality game. *IEEE Transactions on Visualization and Computer Graphics, 18*(4), 626–633. DOI: 10.1109/TVCG.2012.43

McMahan, R. P., Kopper, R., and Bowman, D. A. (2014). Principles for Designing Effective 3D Interaction Techniques. In K. M. Stanney and K. S. Hale (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 285–311). Boca Raton, FL: CRC Press.

McMahan, R.P., Ragan, E.D., Bowman, D.A., Tang, F., and Lai, C. (2015). FIFA: The Framework for Interaction Fidelity Analysis. Tech Report UTDCS-06-15. The University of Texas at Dallas.

Meehan, M., Brooks, F., Razzaque, S., and Whitton, M. (2003). Effects of Latency on Presence in Stressful Virtual Environments. In *Proceedings of IEEE Virtual Reality* (pp. 141–148). Los Angeles, CA.

Meehan, M., Insko, B., Whitton, M., and Brooks, F. P. (2002). Physiological measures of presence in stressful virtual environments. *ACM Transactions on Graphics*. DOI: 10.1145/566654.566630

Melzer, J. E., and Moffitt, K. (2011). *Head-Mounted Displays: Designing for the User*. CreateSpace Publishing.

Merriam-Webster (2015). Merriam-Webster. Retrieved May 8, 2015, from http://www.merriam-webster.com/

Milgram, P., and Kishino, F. (1994). Taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems, E77-D*(12), 1321–1329. DOI: 10.1.1.102.4646

Miller, G. A., and Isard, S. (1963). Some perceptual consequences of linguistic rules. *Journal of Verbal Learning and Verbal Behavior*. DOI: 10.1016/S0022-5371(63)80087-0

Meta Platforms, Inc.
Exhibit 1067
Page 525

Mine, M. (2003). Towards virtual reality for the masses: 10 years of research at Disney's VR studio. In *Proceedings of the workshop on Virtual environments 2003* (pp. 11–17). DOI: 10.1145/769953.769955

Miné, M., and Bishop, G. (1993). *Just-In-Time Pixels*. Technical Report TR93-005, Department of Computer Science, University of North Carolina at Chapel Hill.

Miné, M. R. (1993). *Characterization of End-To-End Delays in Head-Mounted Display Systems*. Technical Report TR93-001, Department of Computer Science, University of North Carolina at Chapel Hill.

Miné, M. R., Frederick P. Brooks, J., and Séquin, C. (1997). Moving Objects in Space: Exploiting Proprioception in Virtual Environment Interaction. In *SIGGRAPH* (pp. 19–26). ACM Press.

Miyaura, M., Narumi, T., Nishimura, K., Tanikawa, T., and Hirose, M. (2011). Olfactory feedback system to improve the concentration level based on biological information. *2011 IEEE Virtual Reality Conference*, 139–142. DOI: 10.1109/VR.2011.5759452

Mlyniec, P. (2013). Motion-Controlled Gaming for Neuroscience Education. Retrieved April 14, 2015, from http://sbirsource.com/sbir/awards/146246-motion-controlled-gaming-for-neuroscience-education

Mlyniec, P., Jerald, J., Yoganandan, A., Seagull, J., Toledo, F., and Schultheis, U. (2011). iMedic: A two-handed immersive medical environment for distributed interactive consultation. In *Studies in Health Technology and Informatics* (Vol. 163, pp. 372–378).

Mori, M. (1970). The Uncanny Valley. *Energy*, *7*(4), 33–35. DOI: 10.1162/pres.16.4.337

Murphy, K. R., and Davidshofer, C. O. (2005). *Psychological Testing: Principles and Applications*. Pearson/Prentice Hall.

Nakayama, K., and Tyler, C. W. (1981). Psychological Isolation of Movement Sensitivity by Removal of Familiar Position Cues. *Vision Research*, *21*, 427–433.

Neely, H. E., Belvin, R. S., Fox, J. R., and Daily, M. J. (2004). Multimodal interaction techniques for situational awareness and command of robotic combat entities. In *IEEE Aerospace Conference Proceedings* (Vol. 5, pp. 3297–3305). DOI: 10.1109/AERO.2004.1368136

Negroponte, N. (1993, December). Virtual Reality: Oxymoron or Pleonasm. *Wired Magazine*. Retrieved from http://archive.wired.com/wired/archive/1.06/negroponte_pr.html

Nichols, S., Ramsey, A. D., Cobb, S., Neale, H., D'Cruz, M., and Wilson, J. R. (2000). *Incidence of virtual reality induced symptoms and effects (VRISE) in desktop and projection screen display systems*. HSE Contract Research Report.

Norman, D. A. (2013). *The Design of Everyday Things, Expanded and Revised Edition. Human Factors and Ergonomics in Manufacturing*. New York: Basic Books. DOI: 10.1002/hfm.20127

Meta Platforms, Inc.
Exhibit 1067
Page 526

Oakes, E. H. (2007). *Encyclopedia of World Scientists*. Infobase Publishing.

Oculus Best Practices. (2015). Retrieved May 22, 2015 from https://developer.oculus.com/documentation/

Olano, M., Cohen, J., Miné, M., and Bishop, G. (1995). Combatting Rendering Latency. In *Proceedings of the ACM Symposium on Interactive 3D Graphics* (pp. 19–24). Monterey, CA.

Pahl, G., Beitz, W., Feldhusen, J., and Grote, K.-H. (2007). *Engineering design: a systematic approach*. *Springer*. DOI: 10.1111/dsu.12130

Patrick, G. T. W. (1890). The Psychology of Prejudice. *The Popular Science Monthly*, *36*, 634.

Pausch, R., Crea, T., and Conway, M. J. (1992). A Literature Survey for Virtual Environments: Military Flight Simulator Visual Systems and Simulator Sickness. *PRESENCE: Teleoperators and Virtual Environments*, *1*(3), 344–363.

Pausch, R., Snoddy, J., Taylor, R., Watson, S., and Haseltine, E. (1996). Disney's Aladdin: first steps toward storytelling in virtual reality. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (pp. 193–203). ACM Press. DOI: 10.1145/237170.237257

Peck, T. C., Seinfeld, S., Aglioti, S. M., and Slater, M. (2013). Putting yourself in the skin of a black avatar reduces implicit racial bias. *Consciousness and Cognition*, *22*(3), 779–787. DOI: 10.1016/j.concog.2013.04.016

Pfeiffer, T., and Memili, C. (2015). GPU-Accelerated Attention Map Generation for Dynamic 3D Scenes. In *IIEE Virtual Reality*. Retrieved from http://gpu-heatmap.multimodal-interaction.org/AttentionVisualizer/GPU_accelerated_Heatmaps.pdf

Pierce, J., Forsberg, A., Conway, M., Hong, S., Zeleznik, R., and Miné, M. R. (1997). Image Plane Interaction Techniques in 3D Immersive Environments. In *ACM Symposium on Interactive 3D Graphics* (pp. 39–44). ACM Press.

Pierce, J. S., Stearns, B. C., and Pausch, R. (1999). Voodoo Dolls?: Seamless Interaction at Multiple Scales in Virtual Environments in Virtual Environments. In *Symposium on Interactive 3D Graphics* (pp. 141–145). DOI: 10.1145/300523.300540

Plato. (380 BC). *The Republic*. Athens, Greece: The Academy.

Pokorny, J., and Smith, V. C. (1986). Colorimetry and Color Discrimination. In K. Boff, L. Kaufman, and J. Thomas (Eds.), *Handbook of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Polonen, M. (2010). *A Head-Mounted Display as a Personal Viewing Device: Dimensions of Subjective Experiences*. University of Helsinki, Finland. Retrieved from https://helda.helsinki.fi/bitstream/handle/10138/19799/aheadmou.pdf?sequence=1

Posner, M. I., Nissen, M. J., and Klein, R. M. (1976). Visual dominance: an information-processing account of its origins and significance. *Psychological Review*, *83*, 157–171. DOI: 10.1037/0033-295X.83.2.157

Meta Platforms, Inc.
Exhibit 1067
Page 527

Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. (1996). The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *ACM Symposium on User Interface Software and Technology* (pp. 79–80). ACM Press. DOI: 10.1145/237091.237102

Poupyrev, I., Weghorst, S., and Fels, S. (2000). Non-isomorphic 3D rotational techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00* (pp. 540–547). DOI: 10.1145/332040.332497

Pratt, A. B. (1916). Weapon. US. US Patent 1183492

Proffitt, D. R. (2008). An Action-Specific Approach to Spatial Perception. *Embodiment, Ego-Space, and Action*, 177–200. Retrieved from http://people.virginia.edu/~drp/reprints/Proffitt_2008_CMU_Chapter.pdf

Proffitt, D. R., Bhalla, M., Gossweiler, R., and Midgett, J. (1995). Perceiving geographical slant. *Psychonomic Bulletin and Review*. DOI: 10.3758/BF03210980

Prothero, J. D., and Parker, D. E. (2003). A Unified Approach to Presence and Motion Sickness. In L. J. Hettinger and M. Haas (Eds.), *Virtual and Adaptive Environments* (pp. 47–66). Boca Raton, FL: CRC Press. DOI: 10.1201/9781410608888.ch3

Provancher, W. (2014). Creating Greater VR Immersion by Emulating Force Feedback with Ungrounded Tactile Feedback. *IQT Quarterly*, 18–21.

Ramachandran, V. S., and Hirstein, W. (1998). The perception of phantom limbs. The D. O. Hebb lecture. *Brain*, *121*(9), 1603–1630. DOI: 10.1093/brain/121.9.1603

Rash, C. E. (2004). Awareness of causes and symptoms of flicker vertigo can limit ill effects. *Flight Safety Foundation-Human Factors and Aviation Medicine*, *51*(2), 1–6. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Awareness+of+Causes+&+Symptoms+of+Flicker+Vertigo+Can+Limit+Ill+Effects#0\nhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Awareness+of+causes+&+symptoms+of+flicker+vertigo+ca

Rasmusson, J. (2010). The Agile Samurai–How Agile Masters Deliver Great Software. *Pragmatic Bookshelf, Año*, 280. Retrieved from http://books.google.de/books?id=KjmXSQAACAAJ\nhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:The+Agile+Samurai:+How+Agile+Masters+Deliver+Great+Software#0

Razzaque, S. (2005). *Redirected Walking*. Department of Computer Science, University of North Carolina at Chapel Hill.

Razzaque, S., Kohn, Z., and Whitton, M. (2001). Redirected Walking. In *Eurographics* (pp. 289–294). Manchester, UK.

Reason, J. T., and Brand, J. J. (1975). *Motion Sickness*. London: Academic Press.

Meta Platforms, Inc.
Exhibit 1067
Page 528

Reeves, B., and Voelker, D. (1993). *Effects of Audio-Video Asynchrony on Viewer's Memory, Evaluation of Content and Detection Ability*.

Regan, D. M., Kaufman, L., and Lincoln, J. (1986). Motion in Depth and Visual Acceleration. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbook of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Regan, M., Miller, G., Rubin, S., and Kogelnik, C. (1999). A Real-Time Low-Latency Hardware Light-Field Renderer. In *Proceedings of ACM SIGGRAPH 99* (pp. 287–290). ACM Press.

Regan, M., and Pose, R. (1994). Priority Rendering with a Virtual Reality Address Recalculation Pipeline. In *Proceedings of ACM SIGGRAPH 94* (pp. 155–162). ACM Press.

Reichelt, S., Haussler, R., Fütterer, G., and Leister, N. (2010). Depth cues in human visual perception and their realization in 3D displays. In *Three Dimensional Imaging, Visualization, and Display 2010* (p. 76900B–76900B–12). DOI: 10.1117/12.850094

Renner, R. S., Velichkovsky, B. M., and Helmert, J. R. (2013). The Perception of Egocentric Distances in Virtual Environments - A Review. *ACM Computing Surveys*, *46*(2), 23:1–40.

Riccio, G. E., and Stoffregen, T. A. (1991). An ecological Theory of Motion Sickness and Postural Instability. *Ecological Psychology*. DOI: 10.1207/s15326969eco0303_2

Riecke, B. E., Schulte-Pelkum, J., Avraamides, M. N., von der Heyde, M., and Bülthoff, H. H. (2005). Scene consistency and spatial presence increase the sensation of self-motion in virtual reality. In *Proceedings of the 2nd symposium on Appied perception in graphics and visualization - APGV '05* (pp. 111–118). DOI: 10.1145/1080402.1080422

Ries, E. (2011). *The Lean Startup*. Crown Business.

Roberts, D. J., and Sharkey, P. M. (1997). Maximising concurrency and scalability in a consistent, causal, distributed virtual reality system, whilst minimising the effect of network delays. In *Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises* (pp. 161–166). DOI: 10.1109/ENABL.1997.630808

Robinson, D. A. (1981). Control of Eye Movements. In V. B. Brooks (Ed.), *The Handbook of Physiology* (Vol. II, Part 2, pp. 1275–1320). Baltimore: Williams and Wilkens.

Rochester, N., and Seibel, R. (1962). Communication Device. US. US Patent 3022878

Rolnick, A., and Lubow, R. E. (1991). Why is the driver rarely motion sick? The role of controllability in motion sickness. *Ergonomics*, *34*, 867–879. DOI: 10.1080/00140139108964831

Salisbury, K., Conti, F., and Barbagli, F. (2004). Haptic rendering: Introductory concepts. *IEEE Computer Graphics and Applications*, *24*(2), 24–32. DOI: 10.1109/MCG.2004.1274058

Meta Platforms, Inc.
Exhibit 1067
Page 529

Samuel, A. G. (1981). Phonemic restoration: insights from a new methodology. *Journal of Experimental Psychology. General*, *110*, 474–494. DOI: 10.1037/0096-3445.110.4.474

Sareen, A., and Singh, V. (2014). Noise Induced Hearing Loss: A Review. *Otolaryngology Online Journal*, *4*(2), 17–25. Retrieved from http://jorl.net/index.php/jorl/article/viewFile/noise_hearingloss/pdf_55

Schneider, N. (2015), The Secret Sauce of Standardization, Gamasutra. Retrieved July 2, 2015, from http://www.gamasutra.com/blogs/NeilSchneider/20150121/234683/The_Secret_Sauces_of_Standardization.php

Schowengerdt, B. T., Seibel, E. J., Kelly, J. P., Silverman, N. L., and Furness III, T. A. (2003, May). Binocular retinal scanning laser display with integrated focus cues for ocular accommodation. In *Electronic Imaging 2003* (pp. 1–9). International Society for Optics and Photonics.

Schultheis, U., Jerald, J., Toledo, F., Yoganandan, A., and Mlyniec, P. (2012). Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3D tasks. In *IEEE Symposium on 3D User Interfaces 2012, 3DUI 2012 - Proceedings* (pp. 117–124).

Sedig, K., and Parsons, P. (2013). Interaction Design for Complex Cognitive Activities with Visual Representations: A Pattern-Based Approach. *AIS Transactions on Human-Computer Interaction*, *5*(2), 84–133. Retrieved from http://aisel.aisnet.org/cgi/viewcontent.cgi?article=1057&context=thci

Sekuler, R., Sekuler, A. B., and Lau, R. (1997). Sound alters visual motion perception. *Nature*. DOI: doi:10.1038/385308a0

Shakespeare, W. (1598). *Henry IV.* Part 2, Act 3, scene 1, 26–31.

Shaw, C., and Green, M. (1994). Two-handed polygonal surface design. In *Proceedings of the 7th annual ACM symposium on User interface software and technology* (pp. 205–212).

Sherman, W. R., and Craig, A. B. (2003). *Understanding Virtual Reality*. Morgan Kaufmann Publishers.

Sherrington, C. S. (1920). *Integrative Action of the Nervous System*. New Haven, CT: Yale University Press.

Siegel, A., and Sapru, H. N. (2014). *Essential Neuroscience* (3rd ed.). Lippincott Williams and Wilkins.

Simons, D. J., and Chabris, C. F. (1999). Gorillas in our midst: Sustained inattentional blindness for dynamic events. *Perception*, *28*(9), 1059–1074. DOI: 10.1068/p2952

Slater, M. (2009). Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, *364*(1535), 3549–3557. DOI: 10.1098/rstb.2009.0138

Meta Platforms, Inc.
Exhibit 1067
Page 530

Slater, M., Antley, A., Davison, A., Swapp, D., Guger, C., Barker, C., Pistrang, N., Sanchez-Vives, M. V. (2006). A virtual reprise of the Stanley Milgram obedience experiments. *PLoS ONE*, *1*(1). DOI: 10.1371/journal.pone.0000039

Slater, M., Pertaub, D. P., Barker, C., and Clark, D. M. (2006). An experimental study on fear of public speaking using a virtual environment. *Cyberpsychology and Behavior*, *9*(5), 627–633. Retrieved from <Go to ISI>://WOS:000241415100018

Slater, M., and Steed, A. J. (2000). A Virtual Presence Counter. *Presence: Teleoperators and Virtual Environments*, *9*(5), 413–434.

Slater, M., and Wilbur, S. (1997). A framework for immersive virtual environments (FIVE): Speculation on the Role of Presence in Virtual Environments. *Presence: Teleoperators and Virtual Environments*, *6*(6). Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=9716&login.asp&site=ehost-live&scope=site

Smith, R. B. (1987). Experiences with the Alternate Reality Kit: An Example of Tension between Literalism and Magic. *IEEE Computer Graphics and Applications*, *7*(8), 42–50. DOI: 10.1109/MCG.1987.277078

So, R. H. Y., and Griffin, M. J. (1995). Effects of Lags on Human Operator Transfer Functions with Head-Coupled Systems. *Aviation, Space and Environmental Medicine*, *66*(6), 550–556.

Spector, R. H. (1990). Visual Fields. In HK Walker, W. Hall, and J. Hurst (Eds.), *Clinical Methods: The History, Physical, and Laboratory Examinations* (3rd ed.). Boston, MA: Butterworths.

Stanney, K. M., Kennedy, R. S., and Drexler, J. M. (1997). Cybersickness is not simulator sickness. In *Proceedings of the Human Factors and Ergonomics Society 41st Annual Meeting* (pp. 1138–1142). DOI: 10.1177/107118139704100292

Stanney, K. M., Kennedy, R. S., and Hale, K. S. (2014). Virtual Environment Usage Protocols. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 797–809). Boca Raton, FL: CRC Press.

Stefanucci, J. K., Proffitt, D. R., Clore, G. L., and Parekh, N. (2008). Skating down a steeper slope: Fear influences the perception of geographical slant. *Perception*, *37*(2), 321–323. DOI: 10.1068/p5796

Steinicke, F., Visell, Y., Campos, J., and Lécuyer, A. (2013). *Human Walking in Virtual Environments*. Springer. DOI: 10.1007/978-1-4419-8432-6

Stoakley, R., Conway, M. J. M. J., and Pausch, R. (1995). Virtual Reality on a WIM: Interactive Worlds in Miniature. In *ACM Confrence on Human Factors in Computing Systems* (Vol. 95, pp. 265–272). DOI: 10.1145/223904.223938

Stoffregen, T. A., Draper, M. H., Kennedy, R. S., and Compton, D. (2002). Vestibular Adaptation and Aftereffects. In K. M. Stanney (Ed.), *Handbook of Virtual Environments* (pp. 773–790). Mahwah, NJ: Lawrence Erlbaum Associates.

Meta Platforms, Inc.
Exhibit 1067
Page 531

Stratton, G. M. (1897). Vision without inversion of the retinal image. *Psychological Review*. DOI: 10.1037/h0075482

Stroud, J. M. (1986). The Fine Structure of Psychological Time. In H. Quastler (Ed.), *Information Theory in Psychology: Problems and Methods* (pp. 174–207). Glencoe, IL: Free Press.

Sutherland, I. E. (1965). The ultimate display. In *The Congress of the Internation Federation of Information Processing (IFIP)* (pp. 506–508). DOI: 10.1109/MC.2005.274

Sutherland, I. E. (1968). A Head-Mounted Three Dimensional Display. In *Proceedings of the 1968 Fall Joint Computer Conference AFIPS* (Vol. 33, part 1, pp. 757–764). Washington, D.C.: Thompson Books.

Taylor, R. M. (2010). Effective Virtual Environments for Microscopy. Retrieved April 23, 2015, from http://cismm.cs.unc.edu/core-projects/visualization-and-analysis/advanced-graphics-and-interaction/eve-for-microscopy/

Taylor, R. M. (2015). *OSVR: Sensics latency-testing hardware*. Retrieved from https://github.com/sensics/Latency-Test/blob/master/Latency_Hardware/Latency_Tester_Hardware.pdf

Taylor, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T. (2001a). VRPN: A Device-Independent, Network-Transparent VR Peripheral System. In *Proceedings of VRST '01* (pp. 55–61). Banff, Alberta, Canada.

Taylor, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T. (2001b). VRPN: A Device-Independent, Network-Transparent VR Peripheral System. Retrieved May 15, 2015, from http://www.cs.unc.edu/Research/vrpn/VRST_2001_conference/taylorr_VRPN_presentation_files/v3_document.htm

Taylor, R. M., Jerald, J., VanderKnyff, C., Wendt, J., Borland, D., Marshburn, D., Sherman, W. R., Whitton, M. C. (2010). Lessons about Virtual Environment Software Systems from 20 Years of VE Building. *Presence: Teleoperators and Virtual Environments*, *19*(2), 162–178.

Treisman, M. (1977). Motion sickness: an evolutionary hypothesis. *Science*, *197*, 493–495. DOI: 10.1126/science.301659

Turnbull, C. M. (1961). Some observations regarding the experiences and behavior of the BaMbuti pygmies. *The American Journal of Psychology*, *74*, 304–308.

Ulrich, R. (1987). Threshold Models of Temporal-Order Judgments Evaluated by a Ternary Response Task. *Perception and Psychophysics*, *42*, 224–239.

Usoh, M., Arthur, K., Whitton, M. C., Bastos, R. R., Steed, A., Slater, M., and Brooks Jr., F. P. (1999). Walking > Walking-in-Place > Flying, in Virtual Environments. In *SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (pp. 359–364). DOI: 10.1145/311535.311589

Meta Platforms, Inc.
Exhibit 1067
Page 532

Valve. (2015). Source Multiplayer Networking. Retrieved May 31, 2015, from https://developer.valvesoftware.com/wiki/Source_Multiplayer_Networking

Van Beers, R. J., Wolpert, D. M., and Haggard, P. (2002). When feeling is more important than seeing in sensorimotor adaptation. *Current Biology*, *12*(10), 834–837. DOI: 10.1016/S0960-9822(02)00836-9

Veer, G. C. van der, and Melguizo, M. del C. P. (2002). Mental Models. In A. Sears and J. A. Jacko (Eds.), *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications* (3rd ed.). Boca Raton, FL: CRC Press.

Viirre, E., Price, B. J., and Chase, B. (2014). Direct Effects of Virtual Environments on Users. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 521–529). Boca Raton, FL: Boca Raton, FL: CRC Press.

Vorlander, M., and Shinn-Cunningham, B. (2014). Virtual Auditory Displays. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 87–114). Boca Raton, FL: CRC Press.

Wake, B. (2003). INVEST in Good Stories, and Smart Tasks. Retrieved April 18, 2015, from http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

Wallach, H. (1987). Perceiving a Stable Environment When One Moves. *Annual Review of Psychology*, *38*, 1–27.

Wallach, H., and Kravitz, J. H. (1965a). Rapid Adaptation in the Constancy of Visual Direction with Active and Passive Rotation. *Psychonomic Science*, *3*(4), 165–166.

Wallach, H., and Kravitz, J. H. (1965b). The Measurement of the Constancy of Visual Direction and of its Adaptation. *Psychonomic Science*, *2*, 217–218.

Wang, J., and Lindeman, R. (2014). Coordinated 3D Interaction in Tablet- and HMD-Based Hybrid Virtual Environments. In *ACM Symposium on Spatial User Interaction*.

Warren, R. M. (1970). Perceptual restoration of missing speech sounds. *Science*, *167*, 392–393. DOI: 10.1126/science.167.3917.392

Warren, W. H., Kay, B. A., Zosh, W. D., Duchon, A. P., and Sahuc, S. (2001). Optic flow is used to control human walking. *Nature Neuroscience*, *4*(2), 213–216. DOI: 10.1038/84054

Webster's New Universal Unabridged Dictionary. (1989). New York: Barnes and Noble Books.

Weinbaum, S. G. (1935, June). Pygmalion's Spectacles. *Wonder Stories*.

Welch, R. B. (1986). Adaptation of Space Perception. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbood of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Welch, R. B., and Mohler, B. J. (2014). Adapting to Virtual Environments. In K. S. Hale and K. M. Stanney (Eds.), *Handbook of Virtual Environments* (2nd ed., pp. 627–646). Boca Raton, FL: CRC Press.

Meta Platforms, Inc.
Exhibit 1067
Page 533

Welch, R. B., and Warren, D. H. (1986). Intersensory Interactions. In K. R. Boff, L. Kaufman, and J. P. Thomas (Eds.), *Handbood of Perception and Human Performance* (Vol. 1). New York: Wiley-Interscience.

Wendt, J. D. (2010). *Real-Walking Models Improve Walking-in-Place Systems*. UNC-Chapel Hill.

Whitton, M. C. (1984). Memory Design for Raster Graphics Displays. *IEEE Computer Graphics and Applications*, *4*(3), 48–65.

Wilkes, C., and Bowman, D. A. (2008). Advantages of velocity-based scaling for distant 3D manipulation. In *ACM symposium on Virtual reality software and technology* (pp. 23–29). DOI: 10.1145/1450579.1450585

Willemsen, P., Colton, M. B., Creem-Regehr, S. H., and Thompson, W. B. (2009). The effects of head-mounted display mechanical properties and field of view on distance judgments in virtual environments. *ACM Transactions on Applied Perception*. DOI: 10.1145/1498700.1498702

Williams, K. (2014). A Wider FOV - A Guide to Virtual Reality Demonstrations. Retrieved April 29, 2015, from http://www.roadtovr.com/wider-fov-special-guide-virtual-reality-demonstrations/

Williams, K., and Mascioni, M. (2014). *The Out-of-Home Immersive Entertainment Frontier: Expanding Interactive Boundaries in Leisure Facilities*. Gower Publishing Limited. Retrieved from http://www.amazon.com/Out—Home-Immersive-Entertainment-Frontier/dp/1472426959/ref=sr_1_1?ie=UTF8&qid=1430409256&sr=8-1&keywords=kevin+williams+immersive

Wingrave, C. A., and LaViola, J. J. (2010). Reflecting on the Design and Implementation Issues of Virtual. *Presence: Teleoperators and Virtual Environments*, *19*(2), 179–195.

Wingrave, C. A., Tintner, R., Walker, B. N., Bowman, D. A., and Hodges, L. F. (2005). Exploring individual differences in raybased selection: strategies and traits. *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* DOI: 10.1109/VR.2005.1492770

Wood, R. W. (1895). The "Haunted Swing" illusion. *Psychological Review*. DOI: 10.1037/h0073333

Yoganandan, A., Jerald, J., and Mlyniec, P. (2014). Bimanual Selection and Interaction with Volumetric Regions of Interest. In *IEEE Virtual Reality Workshop on Immersive Volumetric Interaction*.

Yost, W. A. (2006). *Fundamentals of hearing: An Introduction* (5th ed.). Academic Press.

Young, S. D., Adelstein, B. D., and Ellis, S. R. (2007). Demand characteristics in assessing motion sickness in a virtual environment: Or does taking a motion sickness questionnaire make you sick? In *IEEE Transactions on Visualization and Computer Graphics* (Vol. 13, pp. 422–428). DOI: 10.1109/TVCG.2007.1041

Meta Platforms, Inc.
Exhibit 1067
Page 534

Zaffron, S., and Logan, D. (2009). *The Three Laws of Performance: Rewriting the Future of Your Organization and Your Life*. San Francisco, CA: Jossey-Bass.

Zhai, S., Milgram, P., and Buxton, W. (1996). The influence of muscle groups on performance of multiple degree-of-freedom input. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems Common Ground - CHI '96*, 308–315. DOI: 10.1145/238386.238534

Zimmerman, T. G., Lanier, J., Blanchard, C., Bryson, S., and Harvill, Y. (1987). A hand gesture interface device. In *ACM SIGCHI* (pp. 189–192). DOI: doi:10.1145/30851.275628

Zimmons, P., and Panter, A. (2003). The influence of rendering quality on presence and task performance in a virtual environment. In *IEEE Virtual Reality* (pp. 293–294). DOI: 10.1109/VR.2003.1191170

Zone, R. (2007). Stereoscopic Cinema and the Origins of 3-D Film, 1838-1952. Retrieved from http://books.google.ch/books?hl=en&lr=&id=C1dgJ3-y1ZsC&oi=fnd&pg=PP1&dq=origins+cinema+psychological+laboratory&ots=iUS0TPJ1Vn&sig=S-iW0qoxpCEUil0EySZDoZKNH6c

Zuckerberg, M. (2014). Announcement to Acquire Oculus VR. Retrieved April 8, 2015, from https://www.facebook.com/zuck/posts/10101319050523971

Meta Platforms, Inc.
Exhibit 1067
Page 535

Index

[to be supplied]

Author's Biography

Jason Jerald



Jason Jerald, Ph.D., is Co-Founder and Principal Consultant at NextGen Interactions. In addition to primarily focusing on NextGen Interactions and its clients, Jason is Chief Scientist at Digital ArtForms, serves on multiple advisory boards of companies focusing on VR technologies, coordinates the RTP-VR Enthusiasts Meetup, and speaks about VR at various events throughout the world.

Jason has been creating VR systems and applications for approximately 20 years. He has been heavily involved in over 60 VR-related projects across more than 30 organizations including Valve, Oculus, Virtuix, Sixense, NASA, General Motors, Raytheon, Lockheed Martin, three U.S. national laboratories, and five universities. Jason's work has been featured on ABC's *Shark Tank,* on the Discovery Channel, in the *New York Times,* and on the cover of the MIT Press journal *Presence: Teleoperators and Virtual Environments*. He has held various technical and leadership positions including building and leading a team of approximately 300 individuals, and has served on the ACM SIGGRAPH, IEEE Virtual Reality, and IEEE 3D User Interface Committees.

Jason earned a Bachelor of Computer Science degree with an emphasis in Computer Graphics and Minors in Mathematics and Electrical Engineering from Washington State University. He earned a Masters and a Doctorate in Computer Science from the University of North Carolina at Chapel Hill with a focus on perception of motion and latency in VR. His graduate work consisted of building a VR system with under 8 ms of end-to-end latency; the development of a mathematical model relating latency, head motion, scene motion, and perceptual thresholds; and validation of the model through psychophysics experiments. Jason holds over 20 publications and patents directly related to VR.

Meta Platforms, Inc.
Exhibit 1067
Page 537

The VR Book

Perception and Interaction Design for Virtual Reality

Jason Jerald, *NextGen Interactions*

The VR Book focuses on human-centered design of virtual reality (VR) experiences. Creating compelling VR experiences is an incredibly complex challenge. When VR is done well, these experiences can be brilliant and pleasurable, but when done badly, they can result in frustration and sickness. While there are many causes of bad VR such as the limitations of technology, much is centered on a lack of understanding of human perception, interaction, design principles, and real users.

This book focuses on human elements of VR, such as how users perceive and intuitively interact with various forms of reality, causes of VR sickness, creating useful and pleasing content, and how to design and iterate upon effective VR applications. It is not just for VR designers; it is for the entire team, as they all should understand the basics of perception, VR sickness, interaction, content creation, and iterative design.

Good VR design requires strong communication between human and machine, indicating what interactions are possible, what is currently occurring, and what is about to occur. A human-centered design principle, like lean methods, is to avoid completely defining the problem at the start and to iterate upon repeated approximations and modifications through rapid tests of ideas with real users. Thus, ***The VR Book*** is intended as a foundation for anyone and everyone involved in creating VR experiences including: designers, managers, programmers, artists, psychologists, engineers, students, educators, and user experience professionals.

ABOUT ACM BOOKS

ACM Books is a new series of high quality books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers. ACM Books publications are widely distributed in both print and digital formats through booksellers and to libraries (and library consortia) and individual ACM members via the ACM Digital Library platform.

BOOKS.ACM.ORG · WWW.MORGANCLAYPOOL.COM

ISBN: 978-1-97000-112-9

Meta Platforms, Inc.
Exhibit 1067
Page 538