2017-06-01

An Analysis of Enabling Techniques for Highly-Accessible Low-Cost Virtual Reality Hardware in the Collaborative Engineering Design Process

Joshua Q. Coburn
Brigham Young University

An Analysis of Enabling Techniques for Highly-Accessible Low-Cost Virtual Reality Hardware in the Collaborative Engineering Design Process

Joshua Q. Coburn

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

John L. Salmon, Chair
C. Greg Jensen
Christopher A. Mattson
Derek L. Hansen
Michael D. Jones

Department of Mechanical Engineering

Brigham Young University

ABSTRACT

An Analysis of Enabling Techniques for Highly-Accessible Low-Cost Virtual Reality Hardware in the Collaborative Engineering Design Process

Joshua Q. Coburn
Department of Mechanical Engineering, BYU
Doctor of Philosophy

While there currently exists a great deal of research in the literature demonstrating various engineering applications for virtual reality (VR) and the benefits of these applications, VR adoption has been slow in part because of the high cost and resources required to setup and maintain the hardware for these applications. However, in the last 5 years, a new generation of VR hardware has emerged with cost and resource requirements which are a small fraction of previous hardware.

This work begins with a survey of this newly available hardware summarizing recent advances for providing virtual input to all of the five human senses. The literature review then proceeds to highlight previous research into improving various aspects of the Engineering Design Process by using VR applications. The literature review concludes that given the significantly improved cost to benefit ratio of this new hardware, a tipping point has been reached where companies will see benefits from providing their engineering workforce with general access to VR hardware.

From the conclusions drawn in the literature review, this work proceeds to explore and answer two main questions related to connecting and collaborating via this new VR hardware. The first question seeks to understand the trade-offs between cybersickness and disorientation from different styles of moving users in a collaborative VR environment (CVE). Since a CVE can be much larger than the physical world it is sometimes necessary to move the virtual participant which can cause cybersickness and disorientation. Understanding this trade-off is one key to creating a usable CVE. It is found that many users are willing to experience some mild cybersickness to significantly reduce the amount of disorientation experienced in a CVE. However, a second group of users are not willing to make this trade and hence require the ability to customise the CVE for their preferred trade-off between cybersickness and disorientation.

The second question seeks to understand how a CVE with support for natural gestures can improve communication about complex 3D data over video conferencing which is the current standard for remote collaboration. It is found that such a CVE implemented with the latest low-cost consumer-grade VR hardware can improve communication speed up to 45% while also improving the accuracy of the communication. In addition, it was found that gestures in the CVE were much more effective and natural than mouse gestures in Skype, 93% of participants preferred the CVE over current video conferencing software, and 86% of participants stated they would like to have access to VR tools in their workplace.

Keywords: Virtual Reality, Collaboration, Communication, Engineering Design Process, Gestures, Head Mounted Display

ACKNOWLEDGMENTS

I would like to thank first and foremost my wife and children. Their support and dedication to helping me accomplish this research and degree has been exemplary and without them would not have been possible. I would also like to thank my adviser Dr. John Salmon who spent many hours and late nights with me discussing this research and providing invaluable guidance. In addition, many thanks are owed to my committee members. Dr. Greg Jensen for first seeing and encouraging my potential as a researcher, Dr. Chris Mattson for his advice and feedback, Dr. Derek Hansen for his excellent insight and guidance, and Dr. Mike Jones for his support and guidance. I also thank the college for their financial support of this work.

I would also like to acknowledge the help and support of my extended family. They have provided immense emotional support and advice during the rigors of this degree. In particular I would like to thank Emily Brignone for our long discussions of appropriate statistical methods and Denna and Quinn Coburn and Dave and Julie Thompson for their constant willingness to listen and support.

Finally I acknowledge all my friends in the CAD Lab for their help and support of this work in guiding early research decisions and providing feedback.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

AR	Augmented Reality
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAVE	Cave Automatic Virtual Environment
CSCW	Computer Supported Cooperative Work
COFOR	Common Frame of Reference
CVE	Collaborative Virtual Environment
DDR	Distributed Design Review
DK	Development Kit
DOF	Degree of Freedom
FOV	Field of View
HMD	Head Mounted Display
HRTF	Head Related Transfer Function
IMU	Inertial Measurement Unit
iVP	interactive Virtual Prototype
MR	Mixed Reality
OSVR	Open Source Virtual Reality
SSQ	Simulator Sickness Questionnaire
STEM	Science, Technology, Engineering, and Mathematics
VE	Virtual Environment
VR	Virtual Reality
VRP	Virtual Reality Prototype

CHAPTER 1. INTRODUCTION

1.1 Problem

Existing communication and online meeting software applications are insufficient for conducting long-distance meetings and engineering design reviews with remote participants due to their inability to quickly convey complex data and their hindrance of natural communication [1] which together inhibit collaboration. These meetings typically involve multiple distributed participants; however, unlike in-person meetings only a single person can control the software at a time.

As companies expand globally, engineers are required to work in teams distributed around the world [2]. Although this strategy is usually beneficial, it also introduces numerous challenges into the design process such as cross-cultural communication and time zone differences [3]. One significant challenge is poor collaboration due to the difficulties of achieving natural communication. Currently accepted technologies inhibit effective communication and collaboration between team members [1, 3–6]. One problem with current platforms for holding Distributed Design Reviews (DDR) is that they do not properly convey the many non-verbal expressions [1] that provide additional meaning and context for our verbal communication [7]. Additionally they do not provide a common context [8] in which to discuss complex data such as 3-Dimensional (3D) geometry models or 3D Analysis results. Figure 1.1 illustrates an example collaborative situation in which gestures would be important.

For teams to be able to collaborate effectively, they need to be able to communicate effectively. However, communication is complex, with the audible words and phrases being only a very small portion of the overall communication. Additional communication avenues include hand gestures, facial expressions, lip movement, voice tone, body posture, etc. [8, 9]. Modern voice-only communication systems that are often used for distributed meetings today, such as conference calls, are not able to convey the majority of these communication avenues. More recently,

Figure 1.1: Collaboration around a 3D model.

communication systems have been able to provide live video streams of each user and/or a user's computer screen. While these reopen some previously blocked communication avenues, they still do not provide a single shared context in which gestural interactions become meaningful [8]. These gestural interactions can account for 35% of all actions during a collaborative design activity [9]. An additional communication avenue which is important, though unsupported by current systems, is space [10]. For instance, when entering a room, where we choose to sit or stand sends messages to others in the room. Space provides an environment for our communication and we use this environment's properties (such as size, lighting, acoustics, etc.) and the objects it contains (such as collaborators, tables, chairs, screens, food, etc.) to shape and alter our communication [11].

A proposed solution to this problem is a Collaborative Virtual Environment (CVE) developed to support Engineering Design Reviews. A broad definition of CVEs is given by Snowdon et al. as "a computer-based, distributed, virtual space or set of places ... [in which] people can meet and interact with others, with agents or with virtual objects" [6]. Snowdon notes that CVEs are typically associated with 3D graphical environments which is the intended use-case for this research [6]. Through the CVE, distributed colleagues would be able to see and interact with digital avatars that represent the other participants, models, data, and design artifacts. Avatars would be able to convey some of the more nuanced non-verbal communication by mimicking the non-verbal communication of the user they represent. Users would also be able to see and interact with 3D

geometry and other design artifacts in a more natural way which the other participants would also be able to see. In addition to being able to interact and communicate in a more natural way, the CVE would allow the augmentation of human capabilities by providing abilities and experiences that would be time-consuming, difficult, or even impossible in the physical world [12]. The proposed research will not develop the full CVE but will focus only on exploring the ability of people to view, explain, and comprehend complex 3D models in virtual reality.

1.2 Background

1.2.1 Engineering Design Reviews

The Engineering Design Process is the formal process which engineers use to plan and develop new products and processes while ensuring that they are meeting the project goals and requirements. An important part of the Engineering Design Process is the Design Review. Formal Design Reviews (also referred to as Technical Reviews) are meetings that are held for many purposes including: clarifying assumptions, measuring the state of a project, recieving technical advice from subject matter experts, making team decisions, and reviewing the results of design activities. For example, in the NASA Systems Engineering Process and Requirements Manual NASA has defined 20 different Technical Reviews that each project must convene [13]. The Defense Acquisition University's Project Management Toolkit defines a five step process that each review should follow. These steps include: Plan, Familiarize, Pre-Review, Review, and Follow-Up [14]. These meetings represent a large investment from companies due to the amount of time, money, and number of people involved. Additionally, it is in these meetings that decisions are made which determine a project's future direction and design parameters.

As companies grow and expand their offices it is not uncommon to find engineering teams whose members are located in different offices distributed across the nation or even the world. This distribution can have many advantages such as leveraging local expertise, reducing operating expenses, and allowing companies to "chase the sun." However, having physically distributed teams leads to increased difficulty in collaborating [3, 4]. This difficulty in collaborating can be seen acutely in tasks such as Design Reviews which often require concurrent participation from all members of the group. Since these activities are so critical to the success of a company and gath-

ering distributed team members to a single physical location for each meeting would entail a large expenditure of company resources, many companies have begun using technological platforms to conduct DDRs in hopes of reducing costs while maintaining activity effectiveness and producing acceptable results.

Currently, the most common platform for conducting distributed meetings is the conference call or some variation of it. The two common variations of the conference call are the "video conference call" in which a video stream from each participant or group of participants is streamed to all other participants and the "screen share" in which one participant can share a view of his computer screen with all other participants. While telecommunication companies and design teams have adapted to these technologies, they are based around technology that was developed in the mid 1800s, namely the telephone. Newer technologies have the potential to both improve the experience and meeting effectiveness.

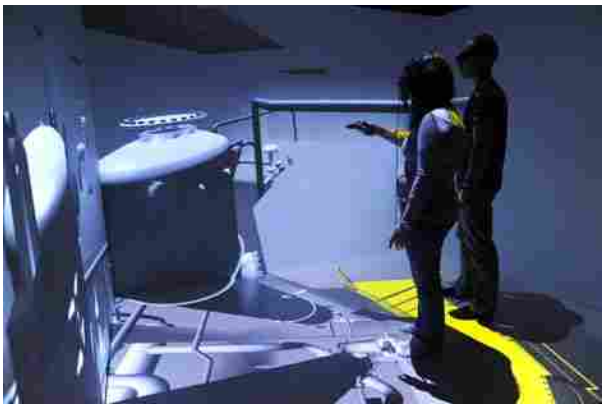
1.2.2 Collaboration

In the 1980s, a new area of research called Computer Supported Cooperative Work (CSCW) began to emerge [9]. This area focuses on understanding how computers can be used as a medium to work collaboratively when the participants are remotely located. One important aspect of CSCW is understanding how humans collaborate when physically co-located in order to better support the same collaboration when distributed. Observational studies of co-located collaboration focused around some artifact (such as design drawings) have highlighted two important features of collaboration that CSCW systems should provide. The first is Gestures and the second is Shared Context [8, 9, 15]. These two elements will be discussed followed by additional pertinent findings regarding CSCW.

Before discussing the importance of gestures in communication it is important to note that with the introduction of touch sensitive devices and motion capture devices, the term "gesture" has gained a new meaning representing a specific motion or motions the device recognizes as input. The type of gesture referred to in this prospectus is the movement of a body part to add additional meaning to communication. Examples of these types of gestures include waving to signal a greeting, and pointing to give context to a statement. In a recent interpersonal communication publication it was found that non-verbal communication is as important if not more important than

our verbal communication [10]. While scholars debate exactly how much of the meaning of a communication is derived from non-verbal communication, the amount generally varies between 65% and 90% [10]. Inside the broad label of non-verbal communication, there are many communication avenues such as Facial Expressions, Eye Behavior, Proxemics, etc. One of these non-verbal channels is Kinesics which covers movement of the body [10]. While it is difficult to quantify exactly how much meaning is derived from each of the various channels in a given situation, one study of group collaboration around a shared design space/artifact found that gestures accounted for slightly more than one third of all the actions performed during the collaboration session [9]. A second study found that communication in collaborative design sessions fell into three categories: storing information, expressing ideas, and mediating interaction. This same study found that the use of gestures fell under all three categories and that conveying gestures would be an important feature of tools intended to support collaborative work [15].

A second important feature of CSCW systems is the Shared Context. John Tang described this requirement well when he said that "gestures [conveyed via a CSCW system] should be kept in context" [15]. In other words, when a person gestures to object A, it should appear to everyone else that he or she is gesturing to object A. An important note is that a shared context can vary in its scope. For instance, screen sharing software allows a shared context of the screen, but any gestures done outside the screen (such as laser pointing) are lost. A common experience, especially when a meeting consists of both co-located and remote participants, is for the presenter to point to an item on the screen and discuss it. The co-located participants can see the gestures, however the remote participants cannot and become partially excluded from the conversation. While it may seem obvious that gestures need to be kept in context, it has not been a major component of many 3D Collaborative Virtual Environments (CVEs) to date. Reasons for this are usually that the CVE simply does not support gestures, or that the CVE is based on Cave Automatic Virtual Environment (CAVE) technology. CAVE technology (shown in Figure 1.2) can only provide a correctly processed picture for a single user. All other users in the same CAVE will see a distorted picture. This distortion ruins the shared context because when a user points to a particular feature the distortion makes it appear to the other users that they are pointing elsewhere [8]. One piece of research has provided a way for two users in a CAVE type setup to see a correctly processed picture



View from inside a CAVE while in-use.

View of the outside of a CAVE showing the space and hardware requirements.

Figure 1.2: An example of a CAVE.

and hence preserve the shared context between the two. This method could possibly be extended to 3 or 4 users, but after that the system would flicker so badly as to render it unusable [8].

In 2001, Steve Benford et al. predicted that the following few years should see CVEs (a type of CSCW system usually based on VR technology) move closer to main stream commercialization [16]. In fact the following years saw quite the opposite happen. Very few of the projects mentioned by Benford are still active and virtual reality is only recently beginning to receive mainstream attention again. Much of this is likely due to the pitfalls associated with CSCW and CVEs. In 1994, Jonathan Grudin identified eight challenges for designing CSCW applications. Important challenges for this area of research include: Disparity in Work-Benefit ratio, Critical Mass and Prisoner's Dilemma problems, and lack of flexibility [17]. He goes on to suggest that the problems require careful consideration when designing a new CSCW application and deploying it [17]. In 2000, Gary and Judith Olson identified more challenges which included technology readiness and

societal readiness [3]. Both technology readiness and societal readiness are interesting since they suggests that both the technology was not ready for mainstream adoption and mainstream society was not ready to adopt collaborative computer applications. However in 2002, Grudin and Palen published a paper suggesting that CSCW applications and societal willingness to adopt was improving [18]. In recent years there has been an explosion of CSCW applications such as the Google Drive apps [19] and Microsoft Office 365 [20]. In engineering, many of the CAD applications have begun adding tools to support collaboration such as Autodesk Fusion 360 [21] and research into this area is ongoing [22,23]. This recent explosion and adoption of CSCW applications, social media, and related technologies suggest that society today is not only willing to adopt collaborative tools, but expects them. Another pitfall of CVEs relates to the use of avatars. Many avatars of the old CSCW applications had to be driven manually and supported only a discrete set of motions. Andrew Mcgrath et al. discussed their observation that users had to focus so closely on "driving" their avatar (moving the avatar around the virtual environment, or manipulating it to show a gesture) that the user's speech slowed down and communication was hindered. Their solution was to automate the avatar (based on what the user was doing on their computer) to allow unimpeded cognition for speech [24].

1.2.3 Virtual Reality & Augmented Reality

Virtual Reality (VR) is a computer-simulated environment that can simulate a physical presence in imagined worlds. Humans experience the real world through five major senses: sight, sound, touch, taste, and smell. Different VR simulations can differ in their immersivity based on which of the five senses they interact with, how much of the real world they replace, and how realistic the provided sensations appear to one's senses. For instance, modern 3D Engineering software could be considered as a very low immersion type of VR. It interacts with only the sense of sight, uses only a small portion of a human's full field of view, and the appearance of the object in the virtual world typically corresponds with what would be expected in shape only. Going to see a movie on an IMAX screen could be considered a more immersive VR experience. It interacts with the sense of both sight and sound. It replaces a very high amount of the real world sensory inputs with virtual ones, and the shape, color, and visual texture are what would be expected if one was having the virtual experience in reality. Both of these examples present the same 2D

image to both eyes and rely on texture, lighting, and relative size to provide us with a sense of 3D space inside of the virtual world. However, when trying to understand the 3-dimensionality of the real world the human brain actually uses two images from slightly different positions along with lighting, texture, relative size, etc. to understand the spatial relationships of the real world. In fact studies have shown that when looking at 3D designs in 3D instead of a 2D computer screen we can more readily pick out errors with designs [25, 26] as well as better understand the design in general [27, 28].

Augmented Reality (AR) is a technology similar to VR. However, instead of replacing the real world with a virtual world, AR overlays additional digital information on the real world [29]. One example of AR is the audio tours commonly found in museums. As visitors move through the museum, they can listen to audio commentary on the various exhibits they are visiting through the use of headphones and a recorded audio tour. However, typically AR is generally considered to include a visual component.

Augmented Reality and Virtual Reality (AR/VR) have been around for at least 50 years. For instance, Head Mounted Displays (HMD) that present a slightly offset image to each eye have existed since at least the 1960s [30] but so far such systems have not see widespread adoption, or research due to the prohibitively high cost of the systems, the cumbersome bulk of the devices, and the low quality images that are typically displayed [31]. Until recently the most well known and commonly used VR hardware was the CAVE which consists of a set of large projector screens and projectors which form a partial or full room. However, the cost of building a CAVE is relatively high. In 2005, NASA commissioned research to build a low cost version of a CAVE. The final iteration cost in the neighborhood of $30,000 [32]. CAVE systems work by tracking the head position of the user, producing a picture for each eye, and then presenting the appropriate picture to the correct eye via the projector screen while blocking the other eye. Since only one person can occupy the correct location from which to view the images, everyone else in the same CAVE sees a distorted view [8]. One research project modified a CAVE system to track two users and produced two sets of pictures for each participant. The picture streams were then interleaved to show both users a correct picture without distortion. The researchers note that this method will only extend to three or four people before flicker becomes unbearable [8]. Thomas Defanti suggests that future CAVEs will transition television screens instead of projectors, but the cost for

this type of CAVE will still be too expensive for companies to deploy widely [33]. These pitfalls have made research into applications of AR/VR technology difficult and has greatly inhibited the commercial adoption of such technologies. However, a proliferation of significantly lower cost consumer grade AR/VR devices is beginning to occur. For example, the recent Oculus Rift™ and Google Glass™ projects have generated much publicity and excitement among the public due to their unprecedented low cost and wider accessibility. Additionally, a wide variety of large tech companies such as Samsung, Sony, Facebook, Epson, and Microsoft, have recently announced their current or impending involvement in this area [34]. HMDs with costs from $300 to $1,500 are now on the market, and smartphones, which many people already own, are being turned into VR/AR devices. Given the cost of CAVEs and the fact that CAVE style systems can generally display an undistorted picture for only one or two people, this places even a cheap CAVE at 10 to 50 times the cost of recent HMDs. This new availability of affordable hardware can enable a variety of applications that were impossible just five years ago such as an improved platform for facilitating design reviews [35].

While VR/AR devices provide a novel way of viewing and interacting with digital data, they are not without their downsides. One of the more troublesome side-effects of AR/VR technology is cybersickness. Cybersickness has symptoms similar to motion sickness such as headache, sweating, disorientation, and nausea [36]. Though it seems similar to motion sickness, researchers still do not fully understand what causes cybersickness. One study has found statistically significant relationships between susceptibility to cybersickness and previous video game experience, previous television and 3D movie use and the use of corrective lenses [37]. LaViola also notes that susceptibility to cybersickness has many contributing factors both technological and individual, hence prediction of susceptibility is difficult [36]. However, given recent research, there is hope that the technical factors can be minimized or eliminated in future devices [38]. AR researchers have also noted the dangers of distraction, present from devices such as cell phones, could become even more pronounced with AR devices [39]. These downsides, along with others not mentioned, show that these new technologies must be applied judiciously.

Table 1.1: Previous Research into Technologically Supported Remote Meetings

System	Features	Findings
Distributed Design Review in Virtual Environments (DDRIVE) 2000	• CAVE based • Discretely animated avatars • Distributed • Live audio	• Network Bandwidth was an issue • Difficult to maintain audio quality
Model, Architecture, and System for Spatial Interaction in Virtual Environments (MASSIVE) 1995 [11, 40]	• 3D room viewed through 2D monitor • Simple avatars • Distributed • Live audio	• Network Bandwidth considerations • Difficulty with turn taking • Avatars unable to convey even most simple gestures
Shared Space 1998 [41, 42]	• Augmented Reality • No need for avatars • Collocated • 3D object viewing at a distance	• Increased communication avenues decreased task time • Participants enjoyed VR experience • Calibration required for shared context • More accurate calibration required for viewing objects closer
Liveboard 1992 [43]	• Shared Digital Whiteboard • Stylus based interaction • Distributed or Collocated	• Poor hardware design can limit usefulness • People were afraid of using it without training
Collaborative Agent Interaction and syncROnization (CAIRO) 2000 [44]	• Conference Call • Static, 2D meeting environment • Distributed • Turn taking features	• Turn taking features decreased meeting time • Need for supporting side conversations

1.2.4 Design Reviews in Virtual Reality

Given the potential benefits of AR/VR technology to 3D design work and CSCW, it is unsurprising to find this technology being applied and researched in many different areas. For instance the use of AR/VR has been explored to enhance the sectors of Architecture [45], Automotive [46], Aerospace [47], Defense [48], Manufacturing [49], Sustainability [50] and Design [51]. Previous work that is related to this research is summarized in Table 1.1.

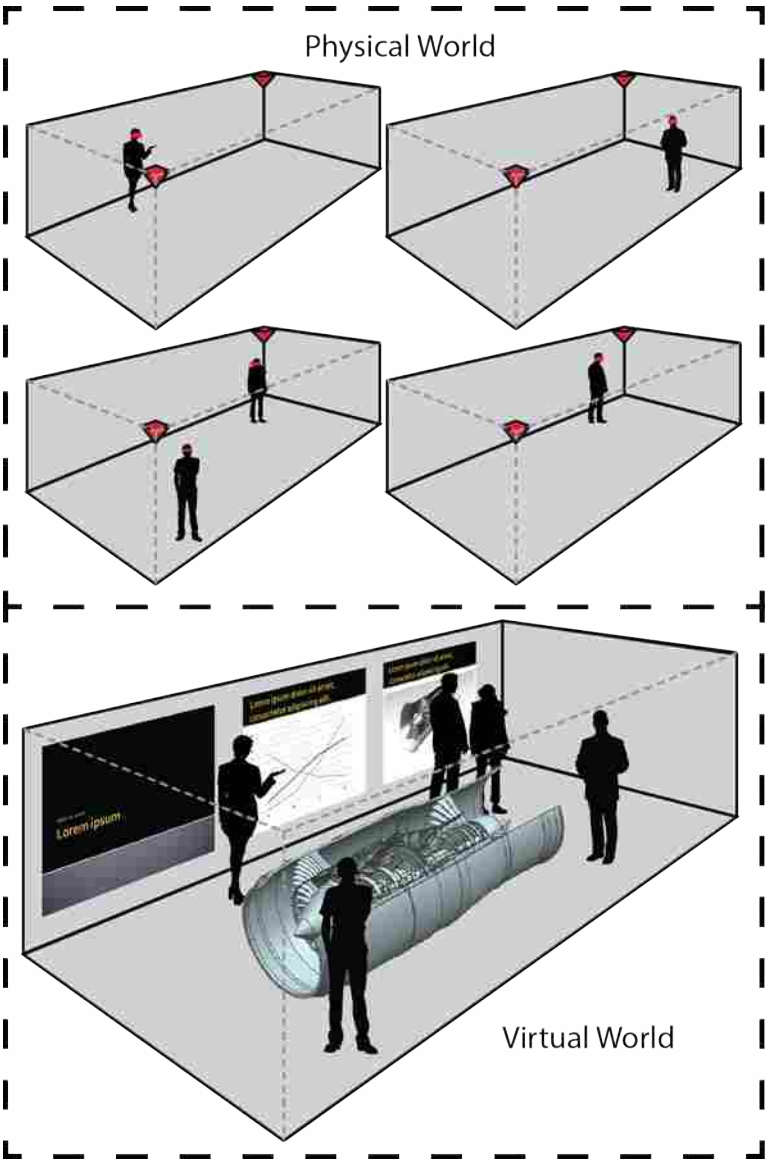


Figure 1.3: An example Distributed Design Review system supported by low-cost hardware in use.

1.3 Long-Term Research Overview

Given the existing research summarized in Section 1.2, we hypothesize that creating a software/hardware system that leverages VR and Motion Capture technology to support Distributed Design Reviews will improve the efficiency and effectiveness of these meetings. This section will describe a long-term vision for this research, some of which is beyond the scope of this dissertation, and the type of system that is envisioned stemming from this line of research. This overview

is presented to provide a context for the questions and research which will be the focus of this dissertation and will be explained in Chapter 2.

1.3.1 Overview

Figure 1.3 shows a virtual DDR taking place using a long-term version of the system. The large picture shows the virtual meeting. Various users are participating in the design review. One is presenting a slide show while another listens, another is inspecting some 3D geometry while listening to the presentation, and two more participants are having a clarifying discussion about a particular point. The presenter can be seen gesturing to a portion of the graph currently being presented. The smaller figures show the corresponding physical users, in their distributed locations, utilizing various pieces of hardware to participate in the virtual meeting.

The full DDR system would consist of virtual meetings which would take place in virtual rooms. Users would utilize various pieces of immersive hardware to make this virtual room a part of their reality. Through the immersive hardware and virtual meeting room, users will participate with the other collaborators in a DDR. Each user would see an avatar in the virtual room for each participant in the meeting. Participants would be able to interact with each other in a variety of ways. They would be able to hold audio conversations with other participants via a conference-call-like audio stream. This audio stream would include special audio processing to adjust the volume level of the audio stream from each participant relative to the listener's position in the virtual room. Users would also be able to use hand, arm, and body gestures to enhance the communication. When a user makes a gesture in the physical world, a motion capture system would capture the motion data. That data would then be used to animate the respective user's virtual avatar in real time so that all other participants in the meeting could see that gesture. Example gestures which might be useful include pointing to a particular part of an artifact, showing relative size, animating a mechanism's operation, or a set of events, etc. Users would also be able to communicate about design artifacts imported into the virtual meeting. These artifacts could include, presentations, text documents, 3D geometries, 2D or 3D analysis results, images, videos, etc. Once imported into the system the artifacts could be presented and discussed in an appropriate manner.

The rest of this section will cover the hardware and software architecture as well as various virtual enhancements that could offer improved performance beyond what is available in co-located meetings.

1.3.2 Hardware

The basic hardware requirements are driven by the requirements of the system. In essence, the system needs to be able to immersively and stereoscopically display the virtual meeting to each participant, the system needs to be able to record motion and position data from each user, and the system needs to be able to stream audio to and from each user. Displaying the virtual meeting to the participant would be accomplished via HMDs. Recording motion and position data from each user would be accomplished by the use of Motion Capture devices. There are many types of motion capture devices each with relative pros and cons. These pros and cons will need to be better understood before selecting hardware for the system. Options include: sensors built into the HMD, room mounted infrared camera, body mounted sensor networks, room mounted camera networks, wands, etc. Audio streaming to and from each participant will be accomplished by the use of headphones and microphones worn by the participant. In prototype systems, hardware that is wired to the computer will provide better performance. However, a final implementation would expect these connections to be wireless or wired to a body mounted computer which is wirelessly powered and networked. This would allow better freedom of motion for participants. Given that hardware which must be worn creates a barrier to use [52], the best option would be sensors that are room-mounted - minimizing the amount of hardware a participant must wear. However, at present, no single room-mounted, consumer-level solution exists to gather all the necessary data to an acceptable level of accuracy.

1.3.3 Software Architecture

The basic software architecture should be client-server since this will allow the server to act as a master and keep all the clients in sync. Each instance of the client software will be responsible for sending data about a particular user (such as position and audio stream) to the server and also for processing and communicating data about other participants to that client's particular user. The

server will be responsible for collecting and distributing the data about all the participants. The server would also be responsible for things such as hosting the meeting, logging in and out of the system, and take on additional roles such as recording the meeting, hosting the artifacts, etc.

1.3.4 Enhancements

The system description until now has been focused on developing a system that would allow distributed participants to, as closely as possible, conduct a design review as if they were all in the same physical room. However, one of the most interesting opportunities associated with VR/AR technology is the possibility of augmenting human capabilities to increase the abilities to perform required tasks. Using VR to create a virtual meeting means that the same rules of physics that apply to the real world can be adjusted at will in the virtual one. This allows for the system to begin to include virtual enhancements that would be difficult, impractical, or even impossible to accomplish in the real world. These enhancements could provide enough positive impact to the meeting productivity and outcomes that using a similar system, even when physically co-located, could prove beneficial to a company. For instance, consider the ability to view and discuss a model of a complex MEMS device as if it were the size of a car (a scale which humans readily relate to and understand) or being able to shrink ourselves to the size of a dust speck to follow a particle flow path through a jet engine. Another example enhancement would be to allow a presenter to see a "line-of-sight" for each participant that indicates where that participant is currently looking - something the real-life presenter currently has only a vague notion of. The presenter could use the lines-of-sight as passive feedback to gauge audience interest and understanding of the presentation and adjust the presentation dialog to match. This feedback could be further enhanced by simultaneously displaying a number of slides before and after the current one, allowing participants to look ahead if the presentation is moving too slowly or look back if they are still struggling with a certain point. With this extra level of feedback the presenter could even more dynamically alter the presentation. Another virtual enhancement would be to allow participants to modify the data presented to explore various scenarios individually and then share their modified scenario with the group if they feel their new results are important. For instance, if a project profitability spreadsheet was under review and a particular participant felt that certain predictions were over optimistic, they could begin modifying those predictions and have only their view of the spreadsheet update. If they

found that the revised predictions seriously impacted the project's potential, they could easily share their view of the spreadsheet with the rest of the group. However, if the revised predictions did little to change the end result, they could choose not to interrupt the meeting by discarding their revisions. Many of these potential system enhancements will need to be explored in subsequent research projects since they would be too large to consider in this dissertation.

CHAPTER 2. RESEARCH OBJECTIVES

2.1 Topic Areas

Given the existing research summarized in Section 1.2, it was hypothesized that creating a software/hardware system to provide a Distributed Design Review experience that allows for basic gesturing as an additional avenue of communication will improve the effectiveness of these meetings. It was also hypothesized that Virtual Reality and motion capture technology have progressed to a point where such a system can be affordably built and deployed. Therefore the primary objective of this research project is to answer fundamental questions surrounding the design and implementation of such a system. First, this research reviews what new low-cost consumer-grade hardware is available, its capabilities, and how it might impact the Engineering Design Process. Second, this research explores and characterizes the trade off between cybersickness and disorientation when moving participants in a VE to a common location in order to create a shared context. Finally, this research demonstrates a prototype CVE with support for gestural communication built with low-cost consumer-grade hardware.

As noted in Section 1.3 this research is intended to be a step towards the larger goal of developing technologies to better collaborate virtually. As such, this prototype system will be designed for use sitting at a desk or other confined space as a means for collaborators to discuss problems and propose changes to a current design. An illustration of an example use-case for the prototype system can be seen in Figure 2.1. In this figure, the left user (blue) can be seen commenting on a certain feature while the right user (purple) asks a question about a second feature. The system allows users to see the 3D design in the virtual world as well as gestures made by other users and themselves in relation to the design. Gestures are conveyed by some type of avatar such as a full body avatar, disembodied hands and wrists, or similar. Additional features may be added to the system as deemed necessary such as rotating the model, zooming the model, animating the head of avatars to show gaze, audio processing to correct for spatial relationship of

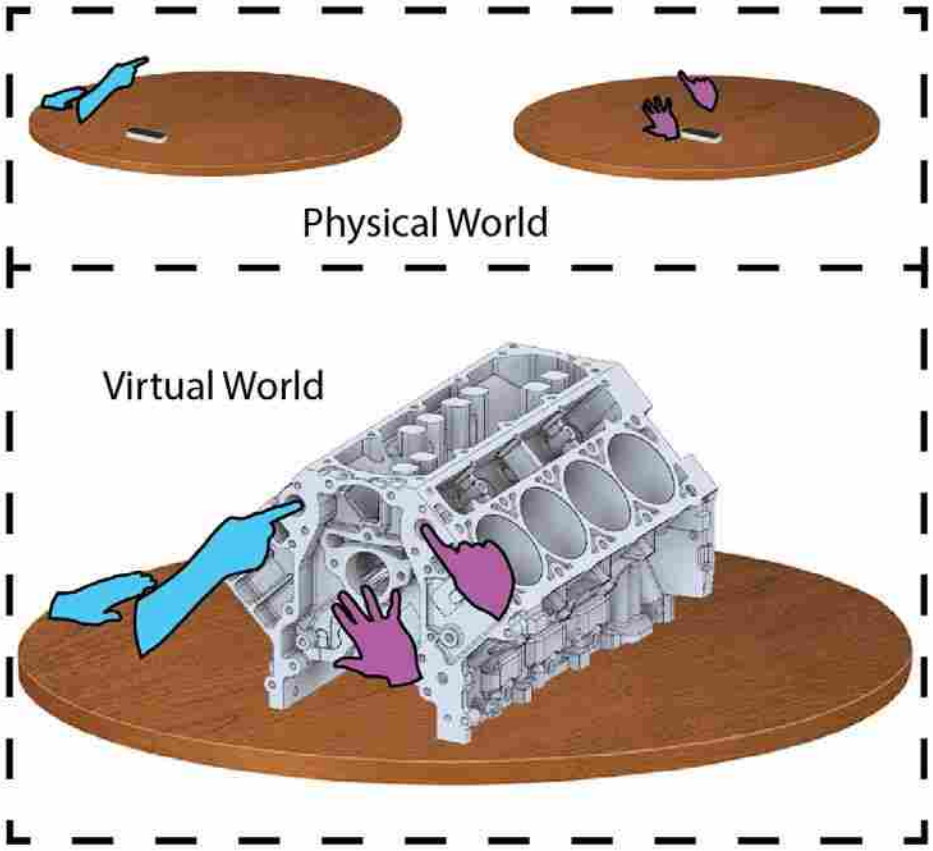


Figure 2.1: Example use-case of Prototype System.

the speaker/listener pair, etc. Table 2.1 shows a breakdown of questions addressed by this research in these three topic areas.

2.1.1 Harware Review

As noted in Section 1.2, during the past five years there has been a proliferation of consumer grade VR hardware that will open up many new research opportunities that were previously prohibitively expensive or not even possible due to inadequate hardware. One of the preliminary goals of this research project is to categorize and overview the various products currently available (and possibly those soon to be released if there is sufficient reliable information) on the consumer market in the areas of VR and Motion Capture. This goal was accomplished via a survey of the literature about available (and soon to be available) hardware.

Additionally, since at least the introduction of CAVE technology in 1992, there has been a significant amount of literature about new applications which leverage VR to improve some aspect

Table 2.1: Research Overview

Research Question	Hypothesis	Research Activity
Hardware Review		
What consumer grade hardware is currently available?	N/A	Lit. Review
What has changed in VR hardware since 2000?	N/A	Lit. Review
Which major areas of the Engineering Design Process have been the focus of previous VR research?	N/A	Lit. Review
How could this new low-cost consumer-grade hardware affect the design process going forward?	N/A	Lit. Review
Shared Context Creation		
What is the trade-off between cybersickness and disorientation for each style of Shared Context creation?	N/A	Experimental Study
How much disorientation results from a teleportation style context creation?	High Amount	Post Exp. Survey
How much cybersickness results from a teleportation style context creation?	Low	Post Exp. Survey
What preference do users have about creation style?	Fly	Post Exp. Survey
What preference do users have about the trade-off between cybersickness and disorientation?	Reduce Disorientation	Post Exp. Survey
Which style of creation allows the quickest identification of the Shared Context location?	Fly	Experimental Study
Which style of creation has the fewest misidentifications of the Shared Context Location?	Fly	Experimental Study
How severe is the cybersickness experienced during Shared Context creation?	Minor	Experimenatl Study
CVE with Gestural Support		
Does a CVE with gesture support allow quicker communication of 3D information?	Yes	Comparison to Legacy
Does a CVE with gesture support allow more accurate communication of 3D information?	Yes	Comparison to Legacy
Do a majority of users prefer the CVE over legacy systems?	Yes	Post Exp. Survey
Do a majority of users find gesturing in a CVE a natural way to communicate 3D information?	Yes	Post Exp. Survey
Do a majority of users find the virtual replication of gestures recognizable and understandable?	Yes	Post Exp. Survey

of the design process. However, these improvements have not permeated the general engineering design community due to the unavailability of the required hardware. This review also looked at this previous literature to surmise which areas of the Engineering Design Process had seen significant focus from VR applications and research, and how a more general availability of the required hardware could impact the Engineering Design Process. This literature review and its conclusions are found in Chapter 3.

2.1.2 Shared Context Creation

As discussed in Section 1.2 the existence of a Shared Context is critical for gestures to make sense. However, since the virtual environment in a CVE can potentially be much larger than the physical space available for the use of VR hardware and users could be located anywhere in the CVE, a method for creating a Shared Context becomes a necessary part of CVE applications. In order to create a Shared Context, a user must be moved to the same virtual area as his collaborators. However, as discussed in 1.2, cybersickness is also an issue with VEs and moving participants' viewpoint in a VE tends to cause cybersickness. In addition, depending on how a user's viewpoint is moved in a VE, the user can become disoriented and lose track of where in the VE they are and what they may be looking at. Both cybersickness and disorientation inhibit the formation of a Shared Context, hence this portion of the research explores the trade-offs between cybersickness and disorientation from different styles of creating a Shared Context. In particular this research looks at the amount of cybersickness, and disorientation caused by Teleportation, Fade-out/Fade-in, Fly, and Manual Shared Context creation styles. In addition to characterizing each creation style's trade-off, users' preferences and the amount of time each style requires to create a Shared Context was recorded. The research pertaining to this topic as well as its results are found in Chapter 4

2.1.3 Prototype CVE with Gesture Support

Many of the cited references in Section 1.2 discuss the importance of gestures in team-based design activities. Common gestures include pointing, raising a hand, demonstrating the working of a mechanism, displaying size or relative location, etc. As such, the final portion of this research consisted of building a prototype CVE with support for gestures and testing its effectiveness for communicating complex 3D data against the currently predominant remote collaboration environment of video conferencing.

This portion of the research demonstrates the feasibility of using this newly available low-cost hardware to create a prototype CVE with gesture support. In addition, it quantifies the differences in the amount of time required to communicate 3D information and the accuracy of the understanding of that information when using the prototype CVE vs video conferencing software.

Users' perefences are also surveyed and presented. The research pertaining to this topic and its results are found in Chapter 5.

2.2 Contributions

The hardware and literature review presented in Chapter 3 has been accepted for publication in the Journal of Computer and Information Science in Engineering. The research presented in Chapter 4 has been submitted for review and publication to Virtual Reality. The research presented in Chapter 5 has been submitted to the Journal of Mechnical Design for review and publication. In addition to these contributions, a provisinal patent disclosure has been filed for the prototype CVE demonstrated in Chapter 5, this research has spun-off several more VR related research projects in the CAD Lab, and portions of this research were highlighted by BYU news[1].

[1] https://news.byu.edu/news/engineering-students-use-virtual-reality-develop-aerospace-and-defense-solutions

CHAPTER 3. HARDWARE & LITERATURE REVIEW

3.1 Introduction

Virtual Reality (VR) hardware has existed since at least the 1960s [30, 38], and more widespread research into applications of VR technology was underway by the late 1980s. By they early 2000s, much of the research had fallen by the wayside, and general interest in VR technology waned. The VR hardware of the time was expensive, bulky, heavy, low resolution, and required specialized computing hardware [25, 30, 53–55]. However, in the last five years, a new generation of hardware has emerged. This new hardware is much more affordable and accessible than previous generations have been, which is enabling research into applications that were previously resource prohibitive. This paper will provide an overview of the specifications of the current generation of hardware, as well as areas of the engineering design process that could benefit from the application of this technology. Section 3.2 will discuss the definition of VR as it pertains to this work. Section 3.3 will discuss current and upcoming hardware for VR. Section 3.4 provides a focused review of the research that has been performed in applying VR to the design process. Section 3.5 will provide a discussion of how the current generation of VR devices may affect research going forward as well as trends seen from the review of the literature. Section 3.5 will also provide some suggestions for research directions based on the concepts reviewed here.

3.2 Definition of Virtual Reality

As discussed by Steuer et al., the term *virtual reality* traditionally referred to a hardware setup consisting of items such as a stereoscopic display, computers, headphones, speakers, and 3D input devices [56]. More recently, the term has been broadly used to describe any program that includes a 3D component, regardless of the hardware they utilize [57]. Given this wide variation, it is pertinent to clarify and scope the term *virtual reality*.

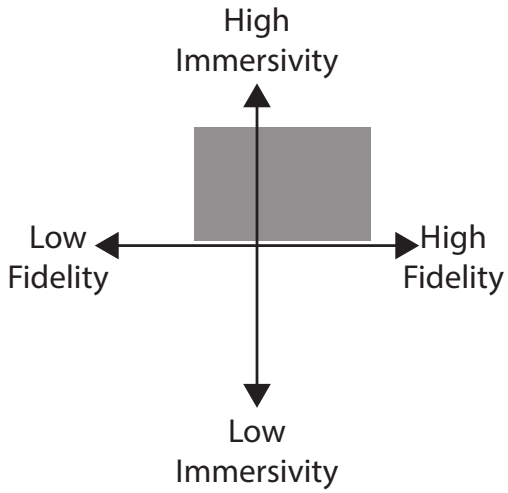


Figure 3.1: Fidelity vs. Immersivity. The shaded portion represents the portion of the VR spectrum under discussion in this paper.

Steuer also proposes that the definition of VR should not be a black-and-white distinction since such a binary definition does not allow for comparisons between VR systems [56]. Based on this idea, we consider a VR system in the light of the VR experience it provides. A very basic definition of a VR experience is the replacing of one or more physical senses with virtual senses. A simple example of this is people listening to music on noise-canceling headphones; they have replaced the sounds of the physical world with sounds from the virtual world. This VR experience can be rated on two orthogonal scales of immersivity and fidelity, see Figure 3.1. Immersivity refers to how much of the physical world is replaced with the virtual world while fidelity refers to how realistic the inputs are. Returning to the previous example, this scale would rate the headphones as low-medium Immersivity since only the hearing sense is affected, but a high fidelity since the audio matches what we might expect to hear in the physical world.

The contrast with augmented reality (AR) should also be noted when discussing VR. While VR seeks to replace physical senses with virtual ones, AR adds virtual information to the physical senses [29]. Continuing the earlier VR example of music on noise canceling headphones, listening to music from a stereo would be an example of AR. In this case the virtual sense (music) is added to the physical sense (sounds from the physical world such as cars). Although there is some overlap between AR and VR technologies and applications, we consider here only technologies for VR,

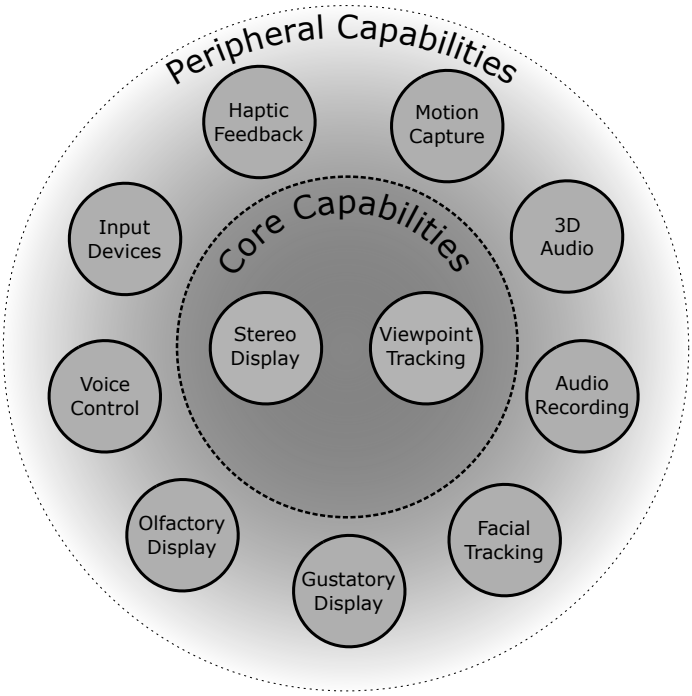


Figure 3.2: Typical Components of a VR Experience. Inner components must be included; outer components are optional depending on the goal of the application.

and we will focus our discussion of applications on VR. For those interested in AR, Kress and Starner [58] provide a good reference for requirements and headset designs.

We also mention here the concept of mixed reality (MR). Current VR technologies are not able to produce high fidelity outputs for all senses. Bordegoni et al. discuss the concept of MR as a solution to this issue. MR combines VR with custom made physical implements to provide a higher fidelity experience [59]. One example is an application to prototype the interaction with an appliance. In this case a user could see the prototype design in VR and at the same time a simple physical prototype would have buttons and knobs to provide the touch interaction with the prototype. In this paper we focus our discussion of application and technologies for pure VR, and as such we will discuss MR only in passing. In addition to the work mentioned previously, Ferrise et al. provide some additional information about MR [60].

In the context of the definition presented, we consider VR experiences that—at a minimum—are high enough fidelity to present stereoscopic images to the viewer's eyes and are able to track a user's viewpoint through a virtual environment as they move in physical space. They must also

be immersive enough to fully replace the user's sense of sight. The dark gray area of Figure 3.1 shows the area under discussion in this paper.

3.3 Virtual Reality Hardware

Various types of hardware are used to provide an immersive, high-fidelity VR experience for users. Given the relative importance the sense of sight has in our interaction with the world, we consider a display system that presents images in such a way that the user perceives them to be 3D (as opposed to seeing a 2D projection of a 3D scene on a common TV or computer screen) in combination with a head tracking system to be the minimum set of requirements for a highly immersive VR experience [30]. This type of hardware was found in almost all VR applications we reviewed, for example [8, 16, 25–28, 30, 45, 49, 51, 52, 55, 61]. This requirement is noted in Figure 3.2 as the core capabilities for a VR experience. Usually some additional features are also included to enhance the experience [56]. These additional features may include motion-capture input, 3D controller input, haptic feedback, voice control input, olfactory displays, gustatory displays, facial tracking, 3D-audio output, and/or audio recording. Figure 3.2 lists these features as the peripheral capabilities. To understand how core and peripheral capabilities can be used together to create a more compelling experience, consider a VR experience intended to test the ease of a product's assembly. A VR experience with only the core VR capabilities might involve watching an assembly simulation from various angles. However, if haptic feedback and 3D input devices are added to the experience, the experience could now be interactive and the user could attempt to assemble the product themselves in VR while feeling collisions and interferences. On the other hand, adding an olfactory display to produce virtual smells would likely do little to enhance this particular experience. Hence these peripheral capabilities are optional to a highly immersive VR experience and may be included based on the goals and needs of the experience. Figure 3.2 lists these core and peripheral capabilities respectively in the inner and outer circles. Devices for providing these various core and peripheral capabilities will be discussed in the following sections.

3.3.1 Displays

The display is usually the heart of a VR experience and the first choice to be made when designing a VR application. VR displays differ from standard displays in that they can present a different image to each eye [30]. This ability to display separate images to each eye allows for presenting slightly offset images to each eye similar to how we view the physical world [62]. When the virtual world is presented this way the user has the impression of seeing a true 3D scene. While the technology to do this has existed since at least the 1960s, it has traditionally been either prohibitively expensive, unwieldy, or a low-quality experience [30,33,55]. VR displays usually fall into one of two groups: Cave Automatic Virtual Experience (CAVE) or Head Mounted Displays (HMDs).

CAVE systems typically consist of two or more large projector screens forming a pseudo-room. The participant also wears a special set of glasses that work with the system to track the participant's head position and also to present separate images to each eye. On the other hand HMDs are devices that are worn on the user's head and typically use half a screen to present an image to each eye. Due to the close proximity of the screen to the eye, these HMDs also typically include some specialized optics to allow the user's eye to better focus on the screen [58, 63]. The following two sections will discuss each of these displays in more detail.

Cave Automatic Virtual Experience

CAVE technology appears to have been first researched in the Electronic Visualization Lab at the University of Illinois [64]. In its full implementation the CAVE consists of a room where all four walls, the ceiling, and the floor are projector screens; a special set of glasses that sync with the projectors to provide stereoscopic images; a system to sense and report the location and gaze of the viewer; and a specialized computer to calculate and render the scenes and drive the projectors [53]. When first revealed, CAVE technology was positioned as superior in most aspects to other available stereoscopic displays [65]. Included in these claims were larger field-of-view, higher visual acuity, and better support for collaboration [65]. While many of these claims were true at the time, HMDs are approaching and rivaling the capabilities of CAVE technology.

The claim about collaboration deserves special consideration. In their paper first introducing CAVE technology, Cruz et al. state, "One of the most important aspects of visualization is communication. For virtual reality to become an effective and complete visualization tool, it must permit more than one user in the same environment" [65]. CAVE technology is presented as meeting this requirement; however, there are certain caveats that make it less than ideal for many scenarios. The first is occlusion. As people move about the CAVE, they can block each other's view of the screen. In general, this type of occlusion is not a serious issue when parts of the scene are beyond the other participant in virtual space although perhaps inconvenient. However, when the object being occluded is supposed to be between the viewer and someone else (in virtual space), the stereoscopic view collapses along with the usefulness of the simulation [53]. A second issue with collaboration in a CAVE is the issue of distortion. Since only a single viewer is tracked in the classic setup, all other viewers in the CAVE see the stereo image as if they were at that location. However, since two people cannot occupy the same physical space and hence cannot all stand at the same location, all viewers aside from the tracked viewer experience some distortion. The amount of distortion experienced is related to the viewer's distance from the tracked viewer [8]. The proposed solution to this issue is to track all the viewers and calculate stereoscopic images for each person. While this has been shown to work in the two-viewer use case [8], commercial hardware with fast enough refresh rates to handle more than two or three viewers does not yet exist.

A more scalable option for eliminating the distortion associated with too many people in the CAVE is to use multiple networked CAVE systems. Information from each individual CAVE can be passed to the others in the network to build a cohesive virtual experience for each participant. This type of approach was demonstrated by the DDRIVE project which was a collaboration between HRL Laboratories and General Motors Research and Development [51]. The downside to this approach is the additional cost and space requirements associated with additional CAVE systems. Each system is typically custom built and prices can range from hundreds of thousands to millions of dollars [66,67]. In 2005 Samuel Miller et al. published research on a low cost, portable CAVE system [32]. Their cost of $30,000 is much more affordable than typical systems, but can still be a significant investment when multiple CAVEs are involved.

Table 3.1: Discrete Consumer HMD Specs (Prices as of 2016)

	Field of View	Resolution per Eye	Weight	Max. Display Refresh Rate	Cost (USD)
Oculus Rift CV1	110 [68, 69]	1080x1200 [68, 69]	440gm [69]	90Hz [68, 69]	$599 [68]
Avegant Glyph	40 [70]	1280 x 720 [70]	434gm [70]	120Hz [71]	$699 [72]
HTC Vive	110 [69, 73]	1200x1080 [69, 73]	550gm [69]	90Hz [69, 73]	$799 [69, 73]
Google Cardboard	Dependent on smart-phone used				$15 [74]
Samsung Gear VR	Dependent on smart-phone used				$99 [75]
OSVR Hacker DK2	110 [63]	1200x1080 [63]	Dependent on configuration	90Hz [63]	$399.99 [63]
Sony Playstation®VR	100 [76]	960x1080 [76]	610gm [76]	120Hz, 90Hz [76]	$399.99 [77]
Dlodlo Glass H1	Dependent on smart-phone used				Unspecified
Dlodo V1	105 [78]	1200x1200 [78]	88gm [78]	90Hz [78]	Expected $559 [79]
FOVE HMD	90-100 [80]	1280x1440 [80]	520gm [80]	70Hz [80]	$399 [81]
StarVR	210 [82]	2560x1440 [82]	380gm [82]	90Hz [83]	Unspecified
Vrvana	120 [84]	1280x1440 [84]	Unspecified	Unspecified	Unspecified
Sulon HMD	Unspecified	Unspecified	Unspecified	Unspecified	$499 [85]
ImmersiON VRelia Go	Dependent on smart-phone used				$139.99 [86]
visusVR	Dependent on smart-phone used				$149 [87]
GameFaceLabs HMD	140 [88]	1280x1440 [88]	450gm [89]	75hz [89]	$500 [89]

Head Mounted Display

As discussed previously, HMDs are a type of VR display that is worn by the user on his or her head. These devices typically consist of one or two small flat panel screens placed a few inches from the eyes. The left screen (or left half of the screen) presents an image to the left eye, and the right screen (or right half of the screen) presents an image to the right eye. Because of the difficulty the human eye has with focusing on objects so close, there are typically some optics placed between the screen and eye that allow the eye to focus better. These optics typically introduce some distortion around the edges that is corrected in software by inversely distorting the images to appear undistorted through the optics. These same optics also magnify the screen, making the pixels and the space between pixels larger and more apparent to the user. This effect is referred to as the "screen-door" effect [35, 90, 91].

In addition to displaying separate images for each eye, these displays typically also track the orientation of the device and consequently the user's head. The orientation of the user's head can then be used as an input control for the VR application allowing the user to turn the camera by turning his or her head. This allows the user to look around the virtual environment just by turning his or her head. This sort of orientation tracking is generally accomplished with an inertial measurement unit (IMU), which generally consists of a three-axis accelerometer and a three-axis gyroscope.

Shortcomings of this type of display can include: incompatibility with corrective eye-wear (although some devices provide adjustments to help mitigate this problem) [92], blurry images due to slow screen-refresh rates and image persistence [93], latency between user movements and screen redraws [94], the fact that the user must generally be tethered to a computer which can reduce the immersivity of a simulation [31], and the hindrance to collocated communication they can cause [52]. The major advantages of this type of display are: its significantly cheaper cost compared to CAVE technology, its ability to be driven by a standard computer, its much smaller space requirements, its ease of setup and take-down (allowing for temporary installations and uses), and its compatibility with many readily available software tools and development environments. Table 3.1 compares the specifications of several discrete consumer HMDs discussed more fully below.

As discussed in Section 3.3.1 the ability to communicate effectively is an important consideration of VR technology. Current iterations of VR HMDs obscure the user's face and especially the eyes. This can create a communication barrier for users who are in close proximity which does not exist in a CAVE as discussed by Smith [52]. It should be noted here that this difference applies only to situations in which the collaborators are in the same room. If the collaborators are in different locations, HMDs and CAVE systems are on equal footing as far as communication is concerned. One method for attempting to solve this issue with HMDs is to instead use AR HMDs which allow you to see your collaborators. Billinghurst et al. have published some research in this area [41,42]. A second method for attempting to solve this issue is to take the entire interaction into VR. Movie producers have used facial recognition and motion capture technology to animate CGI characters with the actor's same facial expressions and movements. This same technology could and has been applied to VR to animate a virtual avatar. Li et al. have presented research supported by Oculus that demonstrates using facial capture to animate virtual avatars [95] and HMDs with varying levels of facial tracking have already been announced and demonstrated [80, 96].

Oculus Rift CV1. The Oculus Rift Development Kit (DK) 1 was the first of the current generation of HMD devices and promised a renewed hope for a low-cost, high-fidelity VR experience and sparked a new interest in VR research, applications, and consumer experiences. The DK1 was first released in 2012 with the second generation (DK2) being released in 2014 and the first consumer version of the Oculus Rift (CV 1) released in early 2016. To track head orientation, the Rift uses a six-degree-of-freedom (DOF) IMU along with an external camera. The camera is mounted facing the user to help improve tracking accuracy. Since these devices are using flat screens with optics to expand the field of view (FOV), they do show the screen-door effect, but it becomes less noticeable as resolution increases.

Steam VR/HTC Vive. The Steam Vive HMD is the result of a collaboration between HTC and Steam to develop a VR system directly intended for gaming. The actual HMD is similar in design to Oculus Rift. The difference though is that the HMD is only part of the full system. The system also includes a controller for each hand and two sensor stations that are used to track the absolute position of the HMD and the controllers in a roughly 4.5m x 4.5m (15ft x 15ft) space. These additional features can make the Steam VR system a good choice when the application requires the user to move around a physical room to explore the virtual world.

Figure 3.3: Images of various HMDs discussed in Section 3.3.1. Top Left to Right: Oculus Rift, Steam VR/HTC Vive, Avegant Glyph. Bottom Left to Right: Google Cardboard, Samsung Gear VR by Oculus, OSVR HDK. (Images courtesy of Oculus, HTC, Avegant, & OSVR.)

Avegant Glyph. The Avegant Glyph is primarily designed to allow the user to experience media such as movies in a personal theater. As such, it includes a set of built-in headphones and an audio only mode where it is worn like a traditional set of headphones. However, built into the headband is a stereoscopic display that can be positioned over the eyes that allows the user to view media on a simulated theater screen. Despite this primary purpose, the Avegant Glyph also supports true VR experiences. The really unique feature is that instead of using a screen like the previously discussed HMDs, the Glyph uses a set of mirrors and miniature projectors to project the image onto a user's retina. This does away with pixels in the traditional sense and allows the Glyph to avoid the screen-door problem that plagues other HMDs. The downside to the Glyph, however, is that it has lower resolution and a much smaller FOV. The Glyph also includes a 9 DOF IMU to track head position.

Google Cardboard. Google Cardboard is a different approach to VR than any of the previously discussed devices. Google Cardboard was designed to be as low cost as possible while still allowing people to experience VR. Google Cardboard is folded and fastened together from a cardboard template by the user. Once the cardboard template has been assembled, the user's

smart-phone is then inserted into the headset and acts as the screen via apps that are specifically designed for Google Cardboard. Since the device is using a smart-phone as the display it can also use any IMU or other sensors built into the phone. The biggest advantage of Google Cardboard is its affordability, since it is only a fraction of the cost of the previously mentioned devices. However, to achieve this low cost, design choices have been made that make this a temporary, prototype-level device not well suited to everyday use. The other interesting feature of this HMD is that since all processing is done on the phone; no cords are needed to connect the HMD to a computer allowing for extra mobility.

Samsung Gear VR. Like Google Cardboard, the Samsung Gear VR device is designed to turn a Samsung smartphone (compatible only with certain models) into a VR HMD. The major difference between these two is the cost and quality. The Gear VR is designed by Oculus and once the phone is attached it is similar to an Oculus Rift. Different from many other HMDs, the Gear VR includes a control pad and buttons built into the side of the HMD that can be used as an interaction/navigation method for the VR application. Also like the Google Cardboard, the Gear VR has no cable to attach to a computer, allowing more freedom of movement.

OSVR Hacker DK2. The Open-Source VR project (OSVR) is an attempt by Razer® to develop a modular HMD that users can modify or upgrade as well as software libraries to accompany the device. The shipping configuration of the OSVR Hacker DK2 is very similar to the Oculus Rift CV1. The notable differences are that OSVR uses a 9 DOF IMU, and the optics use a dual lens design and diffusion film to reduce distortion and the screen-door effect.

Others. Along with the HMDs mentioned above, there are several other consumer-grade HMDs suitable for VR that are available now or in the near future. These include: The Sony Playstation® VR which is similar to the Oculus Rift, but driven from a PlayStation gaming console [97]. The Dlodlo Glass H1 which is similar to the Samsung Gear VR but is compatible with more than just Samsung phones and includes a built in low-latency 9-Axis IMU [98]. The Dlodo V1 which is somewhat like the Oculus Rift, but designed to look like a pair of glasses for the more fashion conscious VR users and is also significantly lighter weight [78]. The FOVE HMD which again is similar to the Oculus Rift, but offers eye tracking to provide more engaging VR experiences [80]. The StarVR HMD is similar to the Oculus Rift with the notable difference of a significantly expanded FOV and consequently a larger device [82]. The Vrvana Totem which is like

the Oculus Rift, but includes built-in pass-through cameras to provide the possibility of AR as well as VR [84]. The Sulon HMD which like the Vrvana Totem includes cameras for AR but can also use the cameras for 3D mapping of the user's physical environment [99]. The ImmersiON VRelia Go which is similar to the Samsung Gear VR but is compatible with more than just Samsung phones [86]. The visusVR which is an interesting cross of the Samsung Gear VR and the Oculus Rift. It uses a smartphone for the screen, but a computer for the actual processing and rendering to provide a fully wireless HMD [87]. The GameFace Labs HMD which is another cross between the Samsung Gear VR and the Oculus Rift. However, this HMD has all the processing and power built into the HMD and runs Android OS [88].

Recent Research in Steroscopic Displays

While currently available and soon-to-be available commercial technologies have been discussed so far, research is ongoing in both HMD and CAVE hardware. Some pertinent research will be highlighted here.

Light Field HMDs. In the physical world, humans use a variety of depth cues to gauge object size and location as discussed by Cruz-Neira et al. [53]. Of the eight cues discussed, only the accommodation cue is not reproducible by current commercial technologies. Accommodation is the the term used to describe how our eyes change their shape to be able to focus on an object of interest. Since, with current technologies, users view a flat screen that remains approximately the same distance away, the user's eyes do not change focus regardless of the distance to the virtual object [53]. Research by Akeley et al. prototyped special displays to produce a directional light field [100]. These light-field displays are designed to support the accommodation depth cue by allowing the eye to focus as if the objects were a realistic distance from the user instead of pictures on a screen inches from the eyes. More recent research by Huang et al. has developed light-field displays that are suitable for use in HMDs [38].

Television-based CAVEs. Currently, CAVEs use rear-projection technology. This means that for a standard size 3m x 3m x 3m CAVE, a room approximately 10m x 10m x 10m is needed to house the CAVE and projector equipment [33]. Rooms this size must be custom built for the purpose of housing a CAVE, limiting the available locations for housing it and adding to the cost of installation. To reduce the amount of space needed to house a CAVE, some researchers have been

exploring CAVEs built with a matrix of television panels instead [33]. These panel-based CAVEs have the advantage of being able to be deployed in more typically sized spaces.

Cybersickness. Aside from the more obvious historical barriers of cost and space to VR adoption, another challenge is cybersickness [101]. The symptoms of cybersickness are similar to motion sickness, but the root causes of cybersickness are not yet well understood [55]. Symptoms of cybersickness range from headache and sweating to disorientation, vertigo, nausea, and vomiting [36]. Researchers are still identifying the root causes, but it seems to be a combination of technological and physiological causes [37]. In some cases, symptoms can become so acute that participants must discontinue the experience to avoid becoming physically ill. It also appears that the severity of the symptoms can be correlated to characteristics of the VR experience, but no definite system for identifying or measuring these factors has been developed to date [55].

3.3.2 Input

The method of user input must be carefully considered in an interactive VR system. Standard devices such as a keyboard and mouse are difficult to use in a highly immersive VR experience [31]. Given the need for alternative methods of interaction, many different methods and devices have been developed and tested. Past methods include wands [102, 103], sensor-gloves [26, 104, 105], force-balls [106] and joysticks [26, 31], voice command [31, 107], and marked/markerless IR camera systems [108–111]. More recently, the markerless IR camera systems have been shrunk into consumer products such as the Leap Motion™ Controller and Microsoft Kinect®. The following sections discuss the various devices used to provide input in a virtual environment. We divide the input devices into two categories: those that are primarily intended to digitize human motion, and those that provide other styles of input.

Motion Capture

In Motion Capture, systems record and digitize movement, human or otherwise. These systems have found applications ranging from medical research and sports training [113, 114] to animation [115] and interactive art installations [116]. Here we are interested in the use of motion capture specifically as an input to a virtual experience. In VR, motion capture is typically used

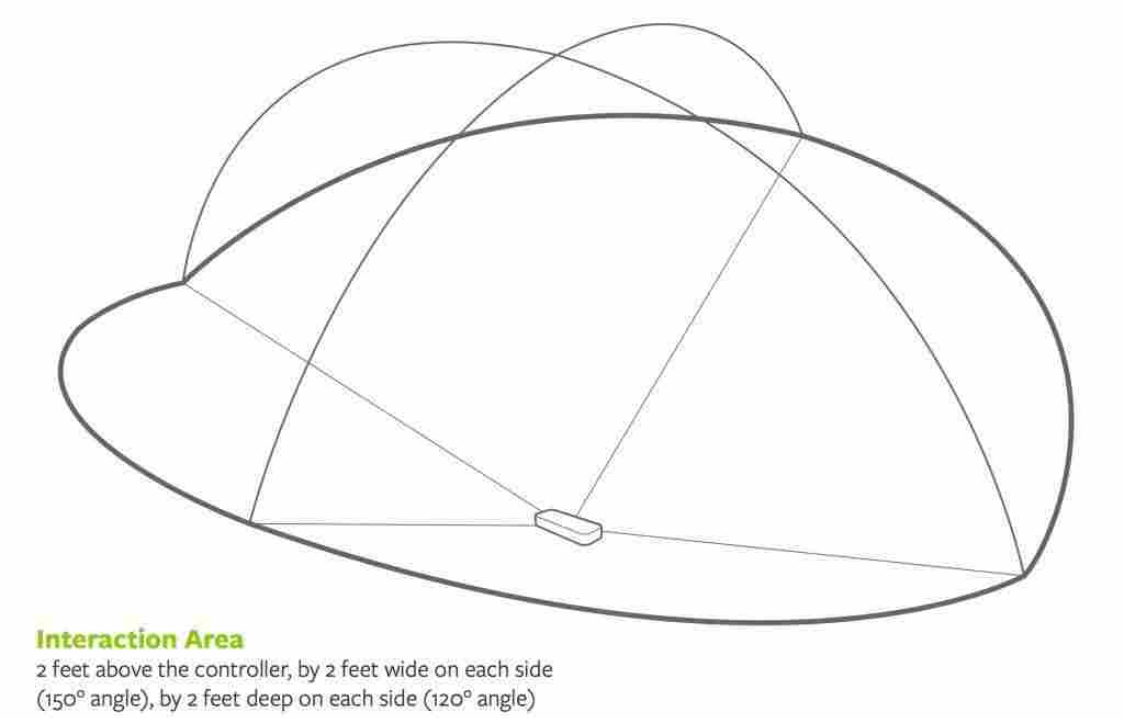


Figure 3.4: Leap Motion™ Controller capture area. Note that newer software has expanded the tracking volume to 2.6 ft (80 cm) above the controller. Figure from the Leap Motion™ Blog [112].

to digitize the user's position and movements. This movement data can then be used directly to animate a virtual avatar of the user allowing them to see themselves in the virtual environment. The movement data can also be analyzed for gestures that can then be treated as special inputs to the system. For instance, the designer may decide to use a fist as a special gesture which brings up a menu. Then, any time the motion capture system recognises a fist gesture, a menu is displayed for the user. While these systems in the past have been large, expensive, and difficult to set up and maintain, in the past five years a new generation of motion capture devices have been released that are opening up potential new applications. Short descriptions of these devices are below.

Leap Motion™ Controller. The Leap Motion™ Controller is an IR camera device approximately 2in. x 1in. x 0.5in. that is intended for capturing hand, finger, and wrist motion data. The device is small enough that it can either be set on a desk or table in front of the user or mounted to the front of an HMD. Since the device is camera based, it can only track what it can see. This constraint affects the device's capabilities in two important ways: First, the view area of the camera is limited to approximately an 8 ft^3 (.23 m^3) volume roughly in the shape of

a compressed octahedron depicted in Figure 3.4. For some applications, this volume is limiting. The second constraint on the device's capabilities is its loss of tracking capability when its view of the tracked object becomes blocked. This commonly occurs when the fingers are pointed away from the device and the back of the hand blocks the camera's view. Weichert et al. and Guna et al. have performed analyses of the accuracy of the Leap Motion™ Controller [117, 118]. Taken together these analyses show the Leap Motion™ Controller is reliable and accurate for tracking static points, and adequate for gesture-based human-computer interaction [118]. However, Guna et. al. also note that there were issues with the stability of the tracking from the device [118] which can cause frustration or errors from the users. Thompson notes, however, that the manufacturer frequently updates the software with performance improvements [35] and since these analysis have been performed, major updates have been released.

Microsft Kinect®. The Microsoft Kinect® is also an IR camera device; however, in contrast to the Leap Motion™ Controller, this device is made for tracking the entire skeleton. In addition to the IR depth camera, the Kinect® has some additional features. It includes a standard color camera which can be used with IR camera to produce full-color, depth-mapped images. It also includes a three-axis accelerometer that allows the device to sense which direction is down, and hence its current orientation. Finally, it includes a tilt motor for occasional adjustments to the camera tilt from the software. This can be used to optimize the view area. The limitations of the Kinect® are similar to that of the Leap Motion™ Controller; it can only track what it has a clear view of and a limited tracking volume. The tracking volume is approximately a truncated elliptical cone with a horizontal angle of 57° and vertical angle of 47° [119]. The truncated cone starts at approximately 4ft. from the camera and extends to approximately 11.5ft. from the camera. For skeletal tracking, the Kinect® also has the limitations of only being able to track two full skeletons at a time; the users must be facing the device, and its supplied libraries cannot track finer details such as fingers. Khoshelham et al. and Dutta et al. evaluated the accuracy of the first version of the Kinect® and found it promising but limited [120, 121]. In 2013, Microsoft released an updated Kinect sensor which Wang et al. noted had improved skeletal tracking which would be further improved by use of statistical models [122].

Intel® RealSense™ Camera. The Intel® RealSense™ is also an IR camera device that can be viewed as a hybrid of the Kinect® and Leap Motion™ Camera. It offers the full color

pictures of the Kinect® with the hand tracking of the Leap Motion™ Controller. It is also important to note that the RealSense™ camera comes in two models: short-range and long-range. The long-range camera (R200) is intended more for depth mapping of medium to large objects and environments. The short-range camera (F200) is intended for indoor capture of hands, fingers, and face. One unique feature the short-range RealSense™ offers is the ability to read facial expressions. However, the RealSense™ cameras have similar issues as the previous two devices including difficulty dealing with occlusion and limited capture volume.

Noitom Perception Neuron®. While the previous discussed motion capture devices all work with IR cameras and image processing, the Perception Neuron® is a very different system. It consists of a group of up to 32 IMUs referred to as Neurons. The IMUs are mounted to the user's body and support various configurations for tailoring the resolution of various areas of the body. The motion capture system streams all the data from the IMUs back to a computer for processing. This data stream can be sent via a WiFi network or a USB cable. Compared to the camera-based systems, the Perception Neuron system does not suffer from occlusion issues and it has a relatively large capture area (limited by the length of the USB cable or strength of the WiFi signal). However, the system is not without its own weaknesses. The most prominent are cost and the user's need to wear a "suit" of sensors. The Perception system costs $1,000–$1,500 depending on the configuration. In contrast, the IR cameras cost $100–$200. Past research has mentioned hardware intrusion as a barrier because of the extra effort to put on and calibrate the hardware, in this case the suit of sensors [52]. An additional weakness is their sensitivity to magnetic interference. Since some of the data collected is orientated by Earth's magnetic field, local magnetic fields such as those generated by computers, electric motors, speakers, and headphones can introduce significant noise when neurons are too close [123].

Controllers

In contrast to the motion capture devices discussed above, controllers aren't primarily intended to capture a user's body movements, but instead they generally allow the user to interact with the virtual world through some 3D embodiment (like a 3D mouse pointer). Many times the controller supports a variety of actions much like a standard computer mouse provides either a left click or right click. A complete treatment of these controllers is outside the scope of this paper

and the reader is referred to Jayaram, et al. [31] and Hayward et al. [124] for more discussion on various input devices. Chapter two of *Virtual Reality Technology* [125] also covers the underlying technologies used in many controllers.

Recently the companies behind Oculus Rift and Vive have announced variants of the wand style controller that blur the line between controller and motion capture. [73,126] These controllers both track hand position and provide buttons for input. The Vive controllers are especially interesting as they work with-in Vive's room-scale tracking system allowing users to move through an approximately 4.5m.x4.5m. (15ft.x15ft.) physical space.

3.3.3 Additional Technologies

The previous sections have discussed viewing and interacting with the virtual world. However, the physical world provides more sensory input than sight. In this section, we will briefly discuss technologies for experiencing the virtual environment through other senses. Given that these areas are entire research fields unto themselves, a thorough treatment of these topics is beyond the scope of this paper and readers are directed to the works cited for more information.

Haptics

Haptic display technology, sometimes referred to as force-feedback, allows a user to "feel" the virtual environment. There are a wide variety of ways this has been achieved. Many times haptic feedback motors are added to the input device used, and thus as the user tries to move the controller through the virtual space, the user will experience resistance when they encounter objects [127]. Other methods include using vibration to provide feedback on surface texture [128] or to indicate collisions [129], physical interaction [129], or to notify the user of certain events such as with modern cell phones and console controllers. Other haptics research has explored tension strings [130], exoskeleton suits [131, 132], ultrasonics [132], and even electrical muscle stimulation [133].

Currently however, commercially available devices are somewhat limited in their diversity and capability. Xia mentions that currently available devices are high-precision, high-resolution devices with small back-drivable forces (i.e. the force a user is required to apply to move the

device), but for many product design applications, they are lacking in workspace size, maximum force of feedback, mechanism flexibility & dexterity, and could use improved back-driveability [134]. For additional information on currently available haptic devices, haptics research in product design, haptic research in product assembly, we refer the reader to works by Xia et al. [134–136]. For more general information on haptics in VR we refer the reader to a study by Burdea [137].

Audio

In addition to localizing objects by sight and touch, humans also have the ability to localize objects through sound [138]. Some of our ability to localize audio sources can be explained by differences in the time of arrival and level of the signal at our ears [139]. However when sound sources are directly in front of or behind us, these effects are essentially nonexistent. Even so, we are still generally able to pinpoint these sound sources due to the sound scattering effects of our bodies and particularly our ears. These scattering effects leave a "fingerprint" on the sounds that varies by sound source position and frequency giving our brains additional clues about the location of the source. This fingerprint can be mathematically defined and is termed the Head-Related Transfer Function (HRTF) [140].

One option for recreating virtual sounds is to use a surround-sound speaker system. This style of sound system uses multiple speakers distributed around the physical space. In using this type of system, the virtual sounds would be played from the speaker(s) that best approximate the location of the virtual source. Since the sound is being produced external to the user, all cues for sound localization would be produced naturally. However, when this system was implemented in early CAVE environments, it was found that sound localization was compromised by reflections (echoes) off the projector screens (walls) [53].

A second option that does not suffer from the echo issue of the surround-sound system is to use specialized audio processing in conjunction with headphones for each user. Since headphones produce sound directly at the ear, all localization cues must be reproduced virtually. While most of the cues are relatively generic, the HRTF is unique to each person and using a poorly matched HRTF to reproduce the localization cues can cause trouble localizing sounds for the participants [140–142]. Thus, for accurately creating virtual sounds getting an accurate HRTF is critical. The standard method for measuring the HRTF of an individual is to place the person in an anechoic

chamber with small microphones in their ears (where headphones would normally be placed) and then one-by-one play a known waveform from various locations around the room and record the signal at the ear [143]. This process is time consuming and unfortunately not very scalable for widespread use [142]; however, research into this area is on going. One study has suggested that it may be possible to pick a HRTF that is close enough from a database of known HRTFs based on a picture of the user's outer ear [141, 144]. Another research group has been studying the inverse of the standard method, whereby speakers are placed in the user's ears and microphones are placed at various locations around the room. This has the advantage that the HRTF can be characterized for all locations at once significantly reducing measurement time [145]. Greg Wilkes of VisiSonics who has licensed this technology hopes to deploy it to booths in stores such as Best Buy where users can have their individual HTRF measured in seconds [146].

Olfactory & Gustatory Displays

While the senses of taste and smell have not received the same amount of research attention as have sight, touch, and audio; a patent granted to Morton Heilig in 1962 describes a mechanical device for creating a VR experience that engaged the senses of sight, sound, touch, and smell [147]. In more recent years, prototype olfactory displays have been developed by Matsukura et al. [148] and Ariyakul et al. [149]. In experiments with olfactory displays Bordegoni et al. showed that an olfactory display could be used to positively improve the level of presence a user in a VR experience perceives [150]. Additionally Miyaura et al. suggests that olfactory displays could be used to improve concentration levels [151]. The olfactory displays discussed here generally work by storing a liquid smell and aerosolizing the liquid on command. Some additionally contain a fan or similar device to help direct the smell to the user's nose. Taste has had even less research than smell, however research by Narumi et al. showed that by combining a visual AR display with an olfactory display they were able to change the perceived taste of a cookie between chocolate, almond, strawberry, maple, and lemon [152].

3.4 Applications of VR in the Design Process

Although different perspectives, domains, and industries may use different terminology, the Engineering Design Process will typically includes steps or stages called opportunity development, ideation, and conceptual design, followed by preliminary and detailed design phases [153]. Often the overall design process will include analysis after, during, or mixed in with, the various design stages followed by manufacturing, implementation, operations and even retirement or disposal [154]. Furthermore, the particular application of a design process takes on various frameworks, such as the classical "Waterfall" approach [155], "Spiral" model [156], or "Vee" model [157], among others [158]. Each model has their own role in clarifying the design stages and guiding the engineers, designers and other individuals within the process to realize the end product. As designs become more integrated and complex the individuals traditionally assigned to the different stages or roles in the design process are required to collaborate in new ways and often concurrently. This, in turn, increases the need for design and communication tools that can meet the requirements for the ever advancing design process.

Finally, while some will consider the formal design stages complete when the manufacturing has begun, a high-level, holistic view of the overall design process from "cradle-to-grave" [159] is most comprehensive and allows the most expansive view for identifying future VR applications. Figure 3.5 shows a summary of the design process described, along with a listing of the applications discussed hereafter. The following sections summarize current applications and briefly suggest additional applications for VR technology. Furthermore, the purpose of the following section is not to provide a comprehensive review of all of the research in this area, but present a limited overview to frame the discussion of how new VR technology can impact the overall design process.

3.4.1 Opportunity Development

It is widely accepted that in order to create successful, user-centered products, designers need to develop empathy for the end user of the product [160]. This empathy is crucial for gaining a clear understanding of the user's needs, and it motivates the designer to design according to those needs [161]. While designers can often develop empathy simply by virtue of shared experiences,

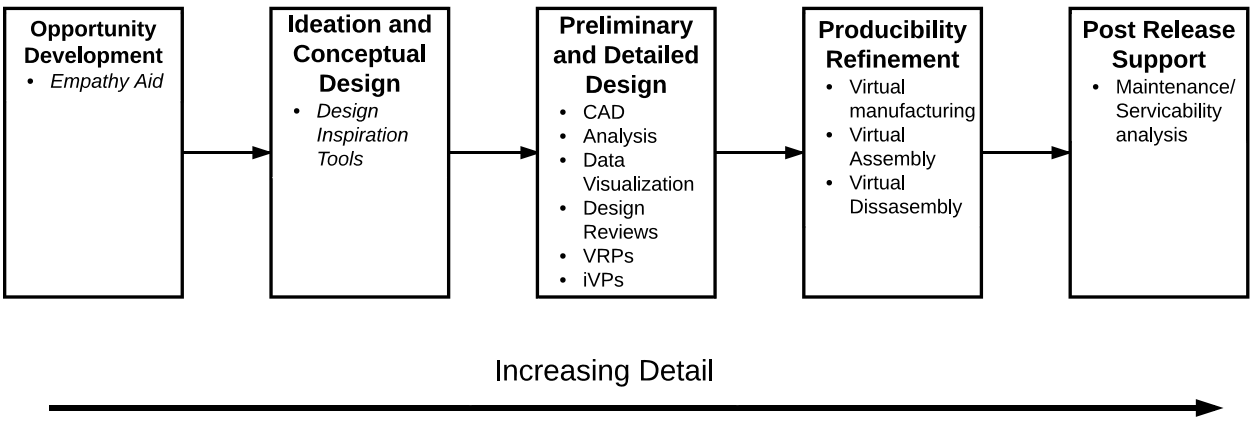


Figure 3.5: Overview of the design process with applications of VR previously explored. Applications in italics represent proposed application rather than existing research.

there are many situations where this approach breaks down, such as a group of young designers working on a product for elderly persons, or a team of male designers designing for pregnant women.

Virtual reality has the potential to provide a novel and effective way of helping designers develop empathy. Recent research has shown that virtual reality can be a powerful tool for creating empathy and even modifying behavior and attitudes. This research has shown that individuals in a virtual environment who are represented by avatars, or virtual representations of themselves, come to have the illusion of ownership over the virtual body by which they are represented [162]. In one experiment, light-skinned participants were shown to exhibit significantly less racial bias against dark-skinned people after the participants were embodied as dark-skinned avatars in virtual reality [163]. A similar study showed that users who were embodied as an avatar with superpowers were more likely to exhibit prosocial behavior after the experiment ended [164].

By leveraging the power of virtual reality, designers could almost literally step into the the shoes of those they are designing for and experience the world through their eyes. A simple application that employs only VR displays and VR videos filmed from the prospective of end users could be sufficient to allow designers to better understand the perspective of those for whom they are designing. Employing haptics and/or advanced controllers also has great potential to enhance the experience. The addition of advanced controllers that allow the designer to control a first person avatar in a more natural way improves immersion and the illusion of ownership over a virtual body [165]. Beyond this, the use of advanced controllers would allow the designer to have

basic interactions with a virtual environment using an avatar that represents a specific population such as young children or elderly persons. The anatomy and abilities of the virtual avatar and environment can be manipulated to simulate these conditions while maintaining a strong illusion of ownership [166–168], thereby giving designers a powerful tool to develop empathy. As in most applications involving human-computer interaction employing haptics would allow for more powerful interactions with the virtual environment, and could also likely be used to better simulate many conditions and scenarios. This technology would have the potential to simulate a wide range of user conditions including physical disabilities, human variability, and cultural differences. Beyond this, designers could conceivably simulate situations or environments such as zero gravity that would be impossible or impractical for them to experience otherwise.

3.4.2 Ideation and Conceptual Design

In the early stages of design, designers and engineers draw upon a diversity of sources for inspiration [169], and indeed all new ideas are synthesized from previous knowledge and experiences [170]. This inspiration comes from both closely related and distantly related or even unrelated sources [171], and it is well understood that both the quality and quantity of ideas generated are positively impacted when designers take time to seek out inspiration [172]. One excellent example of this phenomenon is bio inspired, or biomimetic designs, wherein designs are inspired by mechanisms and patterns found in nature, such as the design of flapping micro air vehicles that mimic flapping patterns of birds [173] or the design of adhesion surfaces patterned after gecko feet [174].

Recent research has shown that technology can facilitate this inspiration process by using computer generated collections of images and concepts that are both closely and distantly related to the subject [175]. Introducing virtual reality to this process has the potential to further facilitate inspiration by giving designers an immersive experience in which they can examine and interact with a huge variety of artifacts. Because these objects exist in a virtual environment, the cost of interacting with these objects is greatly reduced and the quantity of artifacts that designers have access to is dramatically increased. Furthermore, the juxtaposition of artifacts and environments that would not be found together naturally has the potential to provide creative environments that can be superior to existing methods of design inspiration.

Because visual stimulation alone is sufficient to provide significant inspiration to designers [175], an effective VR application targeted at providing design inspiration could be implemented using only low cost VR-displays, reducing both cost and complexity of implementation. The addition of haptics and advanced controllers would likely provide a more interactive experience, allowing designers to touch and handle objects, and would likely aid inspiration. The potential of such an application is supported by recent research that studied the effectiveness of digital mood boards for industrial designers, showing that VR can be used in early stage design to elicit strong emotional responses from designers and facilitate the creative process [176].

3.4.3 Preliminary and Detailed Design

CAD Design

Performing geometric CAD design in a virtual environment has the potential to make 3D modeling both more effective and more intuitive for both new and experienced users. Understanding 3D objects represented on a 2D interface requires enhanced spatial reasoning [177]. Conversely, visualizing 3D models in virtual reality makes them considerably easier to understand and is less demanding in terms of spatial reasoning skills [178], and would significantly reduce the learning curve required by 3D modeling applications. By the same reasoning, using virtual reality for model demonstrations to non CAD users such as management and clients could dramatically increase the effectiveness of such meetings. It should also be noted that there are many user-interface related challenges to creating an effective VR CAD system that may be alleviated by the use of advanced controllers in addition to a VR display.

A considerable quantity of research has been and continues to be conducted in the realm of virtual reality CAD. A 1997 paper by Volkswagen describes various methods that were implemented for CAD data visualization and manipulation, including the integration of the CAD geometry kernel ACIS with VR, allowing for basic operations directly on the native CAD data [179]. A similar kernel-based VR modeler was implemented by Fiorentino et al. in 2002 [180] called SpaceDesign, intended for freeform curve and surface modeling for industrial design applications. Krause et al. developed a system for conceptual design in virtual reality that uses advanced controllers to simulate clay modeling in virtual reality [181]. In 2012 and 2013, De Araujo et al.

developed and proved a system which provides users with a stereoscopically rendered CAD environment that supports input both on and above a surface. Preliminary user testing of this environment shows favorable results for this interaction scheme [182, 183]. Other researchers have further expanded this field by leveraging haptics in order to allow designers to physically interact with and feel aspects of their design. In 2010, Bordegoni et al. implemented a system based on a haptic strip that conforms to a curve, thereby allowing the designer to feel and analyze critical aspects of a design by physically touching them [184]. Kim et al. also showed that haptics can be used to improve upon traditional modeling workflows by using haptically guided selection intent or freeform modeling based on material properties that the user can feel [185].

Much of the research that has been done in this area in the past was limited in application due to the high costs of the VR systems of the 1990s and 2000s. The recent advent of high-quality, low-cost VR technology opens the door for VR CAD to be used in everyday settings by engineers and designers. A recent study that uses an Occulus Rift and the Unity game development engine to visualize engineering models demonstrates the feasibility of such applications [186]; however, research in VR CAD needs to expand into this area in order make the use of low-cost VR technology a reality for day-to-day design tasks.

Analysis

In the same vein as geometric CAD design, virtual reality has the potential to make 3D analysis easier to perform and the results easier to understand, especially for non-analysts [187]. By making the geometry easier to understand, VR can facilitate preprocessing steps that require spatial reasoning, such as mesh repair and refinement. VR can also facilitate understanding and interpretation of analysis results not only by providing a more natural 3D environment in which to view the results, but it can also provide new ways of interacting with the results.

Significant progress has been made in this field in the last 25 years and researchers have explored a range of applications, from simple 3D viewers to haptically enabled environments that provide new ways of exploring the data. A few early studies proved that VR could be used to simulate a wind tunnel while viewing CFD results [188, 189]. Bruno et al. also showed that similar techniques can be used to overlay and view analysis results on physical objects using augmented reality [190]. In 2009, Fiorentino et al. expanded on this by creating an augmented reality sys-

tem that allowed users to deform a physical cantilever beam and see the stress/strain distribution overlaid on the deformed beam in real time [191, 192]. A 2007 study details the methodology and implementation of a VR analysis tool for finite element models that allows users to view and interact with FEA results [193]. Another study uses neural nets for surrogate modeling to explore deformation changes in an FEA model in real time [194]. Similar research from Iowa State University uses NURBS-based freeform deformation, sensitivity analysis, and collision detection to create an interactive environment to view and modify geometry and evaluate stresses in a tractor arm. Ryken and Vance applied the system developed to an industry problem and found that the system allowed the designer to discover a unique solution to the problem [195].

Significant research has also been performed in applying haptic devices and techniques to enhance interaction with results from various types of engineering analyses. Several studies have shown that simple haptic systems can be used to interact with CFD data and provide feedback to the user based on the force gradients [196,197]. Ferrise et al. developed a haptic finite element analysis environment to enhance the learning of mechanical behavior of materials that allows users to feel how different structures behave in real time. They also showed that learners using their system were able to understand the principles significantly faster and with less errors [198, 199]. In 2006, Kim et al. developed a similar system that allows users to explore a limited structural model using high degree of freedom haptics [185].

One trend that we can observe from the research in this field is that it has focused on high-level applications of VR to analysis, such as viewing results and low fidelity interactive analysis. This type of application makes sense in the context of the expensive VR systems that have existed in the past; however, with the advent of modern inexpensive VR headsets, lower-level applications that focus on the day-to-day tasks of analysis become feasible, opening a new direction for research.

Data Visualization

The notion of using virtual reality as a platform for raw data visualization has been a topic of interest since the early days of VR. Research has shown that virtual reality significantly enhances spatial understanding of 3D data [200]. Furthermore, just as it is possible to visualize 3D data in 2D, virtual reality can make interfacing with higher dimensional data more meaningful. A 1999

study out of Iowa State shows that VR provides significant advantages over 2D displays for viewing higher dimensionality data [201]. A more recent study found that virtual reality provides a platform for viewing higher dimensional data and gives "better perception of datascape geometry, more intuitive data understanding, and a better retention of the perceived relationships in the data" [202]. Similar to how analysis results can be explored in virtual reality, haptics and advanced controllers can be used to explore the data in novel ways [178, 203]. Brooks et al. also proved this in 1990 by creating a system that allows users to explore molecular geometry and their associated force fields that allowed chemists to better understand receptor sites for new drugs [204].

Design Reviews

Design reviews are a highly valued step in the design process. Many of the vital decisions that decide the final outcome of a product are made in a design review setting. For this reason, they have been and continue to be an attractive application for virtual reality in the design process, and are one of the most common applications of VR to engineering design [205]. Two particularly compelling ways in which virtual reality can enhance design reviews are by introducing the possibility for improved communication paradigms for distributed teams and enhanced engineering data visualization. In this way, most VR design review applications are extensions of collaborative virtual environments (CVE). CVEs are distributed virtual systems that offer a "graphically realised, potentially infinite, digital landscape" within which "individuals can share information through interaction with each other and through individual and collaborative interaction with data representations" [206].

A number of different architectures have been suggested for improving collaboration through virtual design reviews [205, 207, 208]. Beyond this, various parties have researched many of the issues surrounding virtual design reviews. A system developed in the late 1990s called MASSIVE allows distributed users to interact with digital representations (avatars) of each other in a virtual environment [40, 209]. A joint project between the National Center for Supercomputing Applications, the German National Research Center for Information Technology, and Caterpillar produced a VR design review system that allows distributed team members to meet and view virtual prototypes [210]. A later project in 2001 also allows users to view engineering models while also representing distributed team members with avatars [211].

It should be noted that considerable effort has also been expended in exploring the potential for leveraging virtual and augmented reality technology to enhance design reviews for collocated teams. In 1998, Szalavári et al. developed an augmented reality system for collocated design meetings that allows users to view a shared model and individually control the view of the model as well as different data layers [212]. A more recent study in 2013 compares the immersivity and effectiveness of two different CAVE systems for virtual architectural design reviews [213]. Other research has examined the use of CAVE systems for collocated virtual design reviews [213]. In 2007, Santos et al. further opened this space by proposing and validating an application for design reviews that can be carried out across multiple VR and AR devices [205]. Yet other research has shown that multiple design tools, such as interactive structural analysis, can be integrated directly into the design review environment [192].

While design reviews are a common and popular application of virtual reality to collaborative engineering, the techniques discussed above could be applied to enhance engineering collaboration between distributed team members in many situations, including both formal and informal meetings.

Virtual Reality Prototype

One of the primary elements in engineering, and in design in general, is to evaluate the merit of a given design and identify weak points that need to be refined. Engineers and designers use a wide array of tools to accomplish this task including mathematical models, finite element models, and prototypes.

Another technique that has been the subject of considerable research since the advent of modern CAE tools is virtual prototyping [214]. The term *virtual prototype* has been used in the literature to mean a staggering number of different things; however, Wang defines a virtual prototype as "a computer simulation of a physical product that can be presented, analyzed, and tested from concerned product life-cycle aspects such as design/engineering, manufacturing, service, and recycling as if on a real physical model" [215]. Many have also used the term *virtual prototype* to imply the involvement of virtual reality technologies, but in an effort to promote specificity and clarity, we propose a new term: *Virtual Reality Prototype* (VRP), which refers to virtual prototypes for which virtual reality is an enabling technology. VRPs are an especially compelling branch of

VPs due to the fact that they proffer a set of tools that lend themselves to creating rich human interaction with virtual models, namely stereoscopic viewing, real time interaction, naturalistic input devices, and haptics. In cases where VRP's are specifically used in conjunction with haptics or advanced controllers in order to prototype the human interaction with the virtual models, Ferrise et al. have proposed the term *interactive Virtual Prototype* (iVP), which we employ here to define this subset of VRPs [216].

Aesthetic Evaluation. Because virtual reality enables both stereoscopic viewing of 3D models and an immersive environment in which to view them, using VRPs can provide a much more realistic and effective platform for aesthetic evaluation of a design. Not only does VR allow models to be rendered in 3D, but they can also be viewed in a virtual environment that is similar to one in which the product would be used, thereby giving better context to the model. Furthermore, VR can enable users to view the model at whatever scale is most beneficial, whether it be viewing small models at a large scale to inspect details, or viewing large models at a one-to-one scale for increased realism. Research at General Motors has found that viewing 3D models of car bodies and interiors at full scale provides a more accurate understanding of the car's true shape than looking at small-scale physical prototypes [52]. Another study at Volvo showed that using VR to view car bodies at full scale was a more effective method for evaluating the aesthetic impact of body panel tolerances than using traditional viewing methods [217].

Usability and Ergonomics. The unique input methods and haptic controllers proffered by VR technology provide an ideal platform for simulating and evaluating product-user interactions in a virtual environment. By using hapics and advanced input devices, these iVPs can be used to evaluate the usability and ergonomics of a design. iVPs can enable users to pick up, handle, and operate a virtual model. Based on the evaluation of the iVP, changes can be made to the model and the iVP can be re-evaluated to iterate on a design far more quickly than physical prototypes permit. In 2006, Bordegoni et al. showed that haptic input devices could be used to evaluate the ergonomics of physical control boards [218]. In 2013, Bordegoni et al. extended this research by further defining iVPs and presenting a methodology for designing interaction with these iVPs. In these papers they also presented several user-based case studies that show that iVPs can be used to simulate physical prototypes to an acceptable degree of realism [216, 219]. In 2010, Bruno et al. corroborated these finding by showing that advanced input devices can be an effective method of

evaluating and improving the usability of physical user interfaces represented through VRPs [220]. As mentioned in section 2, another interactive Virtual Prototyping technique that has shown potential is mixed prototyping. A mixed prototype is an integrated and co-located mix of generally low-fidelity physical and high-fidelity virtual components that allows users to interact with simple physical objects that are digitally overlaid or replaced with virtual representations [59, 221–225]. This mix of physical and virtual components can allow for rapid and low cost evaluation of concepts that have good visual fidelity.

Early Stage VRPs. Due to the high cost of building detailed physical prototypes, they are often not used in the early stages of design, such as concept selection and early in the detailed design phase. The cost of creating VRPs however can be much lower because they can be based on CAD geometry of any fidelity. Furthermore, parametric CAD models can be used to quickly explore a wide range of concepts and variations using a single model. Once the CAD geometry has been created, a variety of techniques including those described above can be used to evaluate the model. Consequently, VRPs can enable a more complete evaluation of concepts and models earlier in the design process. In keeping with with this concept, Noon et al. created a system that allows designers to quickly create and evaluate concepts in virtual reality [226].

Market Testing. By putting VRPs in the hands of a market surrogate, all of the benefits that VRPs provide could be realized for market testing including reduced cost, increased flexibility, and the ability to test earlier in the design process. Additionally VRPs can enable novel approaches to market testing. For example, leveraging parametric CAD models could allow market surrogates to evaluate a large number of design variations rather than a single prototype. Alternatively, users could be given a series of VRPs that vary incrementally from a nominal model. After examining and/or interacting with each model, the user could either toss the VRP to the right or to the left based on whether they felt that the VRP was better or worse than the last one they were presented with. In this way, VRPSs could be used to perform a human-guided optimization on design aspects that are difficult to quantify such as aesthetics or ergonomics.

3.4.4 Producibility Refinement

In an effort to continually reduce costs and time to market, engineers and designers have put increased focus on design for manufacturing and design for assembly and the integration of

these activities earlier and earlier in the design process. Knowing this, it comes as no surprise that leveraging virtual reality for these processes has been an area of considerable research over the last 25 years. One of the greatest strengths of virtual manufacturing and assembly is that it is well suited toward analyzing the human factors in manufacturing and assembly. Through VR, designers can closely simulate the manufacturing and assembly steps required for a product using iVPs, and therefore quickly iterate to refine the manufacturability of the design. In this sense, haptics and advanced controllers are well suited to virtual manufacturing and assembly as they allow for more natural interaction with virtual geometry.

Research in this field has ranged from early systems that used positional constraints to verify assembly plans [227] and VR-based training for manufacturing equipment [228] to the exploration of integrated design, manufacturing and assembly in a virtual environment [229], and haptically enabled virtual assembly environments [230].

In the same vein, researchers have explored the extension of virtual reality techniques to design for disassembly and recycling [231, 232]. Using many of the same techniques, designers can evaluate the ease of disassembly of a product early in the design process, and therefore more easily design eco-friendly products.

As mentioned above, this field of research is extensive, and treatment of its full breadth and depth is beyond the scope of this paper. For a more complete exploration of this topic, we refer the reader to Seth et al. [233] for a recent survey of virtual assembly and Choi et al. [234] for a recent survey of virtual manufacturing.

3.4.5 Post Release Support, Repair/Maintenance

As systems grow larger, more complex, and more expensive, maintainability becomes a serious concern, and design for maintainability becomes more and more difficult [235]. One of the issues that exacerbates this difficulty is that serious analysis of the maintainability of a design cannot be performed until high fidelity prototypes have been created [236]. One way in which designers have attempted to address this issue is through simulated maintenance verification using CAD tools [237]. This approach, however, is limited by the considerable time required to perform the analysis, and the lack of fidelity when using simulated human models.

As with design for assembly, the use of VR has the potential to allow designers to do detailed maintainability and serviceability studies earlier in the design process. Using haptics and advanced controllers can allow designers to simulate maintenance scenarios and then allow them to interact with geometric assembly models in a natural way and thereby evaluate and refine the serviceability of the design.

Many researchers have explored the application of virtual reality to design for maintainability. In 1999, Gomes de Sa and Zachmann suggested a combined VR assembly and maintenance verification tool [238]. In 2004 Borro et al. implemented a compelling system for maintenance verification of jet turbine engines using virtual reality and haptic controllers [239]. Peng et al. implemented a system that allows product designers and maintainability technicians to collaborate and evaluate maintenance tasks in a virtual environment [240].

3.5 Discussion

From the foregoing explorations, the authors identify a few key themes that should be underscored and recognized as potential avenues to further develop and implement VR in the various stages of the overall design process. Until recently, VR technology has been applied to the "high-cost" activities defined as events, meetings, and other situations where a key set or large group of decision makers gather for investment decisions and/or decisions about the continuation of significant project resources. This is, in part, to justify the high expense of legacy VR systems. More recently lower cost design activities are now feasible with the corresponding lower cost of VR technology. Another theme is the evidence of realizable and potential impacts that VR can have on the design process. VR is being applied to many more smaller tasks in the design process and initial studies, as explored in the previous sections, suggest a high probability of continuing and expanding this technology to reap the benefits. A third theme is the potential to leverage the trade between current VR capability and cost. At one tenth the cost, or even lower, current generation VR systems are approaching the experience, resolution, and benefit of the larger more complex systems. In the future, this capability gap may further shrink while associated costs may also decrease. The following paragraphs will further discuss and highlight these themes in the context of improving the design process using VR technology.

For years CAVE systems have been considered the gold standard for VR applications. However, because of the capital investment required to build and maintain a CAVE installation, companies rarely have more than one CAVE if any. This significantly limits access to these systems and their use must be prioritized for only select activities. This situation could be considered analagous to the mainframe computers of the 60's & 70's. While these mainframes improved the Engineering Design Process and enabled new and improved designs, it was not until the advent of the PC that computing was able to impact day-to-day engineering activities and make previously unimagined applications commonplace. In a similar fashion, we suggest that this new generation of low-cost, high-quality VR technology has the potential to bring the power of VR to day-to-day engineering activities. Much like the PCs we can expect initial implementations and applications to be somewhat crude and unwieldy while the technology continues to grow and better practices emerge, but the ultimate impact is likely only limited by the imagination of engineers and developers.

As noted previously, current VR systems in industry are unavailable for all but the highest priority tasks. This limits both the potential applications and benefits of the CAVE system and limits a CAVE's cost to benefit ratio. HMDs are currently undergoing significant improvements and are fast approaching CAVE systems in terms of the fidelity and immersivity of the experience they can provide. Additionally, even if a few HMDs are required to serve an equivalent number participants, the capital investment is a small fraction of what is required for a CAVE system. This means that HMD systems have the potential to provide a much better cost to benefit ratio even when used only for the same applications traditionally requiring a CAVE. However, continuing the same mainframe/PC analogy from before, PCs were not invented to initially enable better communication as email was of limited use until a sufficient number of people had consistent access to it on their PC. Now, the vast majority of communication happens digitally, and communication technology is evolving beyond email to social media. None of this would have been possible with only limited-access mainframe computers. Similarly, when a larger number of people have access to VR tools for their daily tasks, new use-cases can be explored which currently are unimagined. While the benefits of these yet-unimagined use cases can not be quantified, they will further improve the cost/benefit ratio of VR HMDs.

Another trend that we observe from the review of the research that has been performed to date is that the majority of what has been done focuses on the mid to later stages of the design process, and that very little has been done to enhance the very early stages of design (e.g. Opportunity Development and Ideation). This trend can clearly be observed in Figure 3.5. While the applications of VR technologies to the early stages of design are perhaps less obvious, there is certainly room for the research to expand in this direction, as is indicated by the potential applications described in sections 3.4.1 and 3.4.2.

Finally, in examining the research that has been performed in this field, we observe the trend that a significant portion of the research merely presents a methodology or details the implementation for a novel or improved application of virtual reality to the design process. The minority of the studies considered performed some form of validation that the application developed was better than existing tools, and indeed only a small minority of studies included a rigorous analysis of the application developed. While this is understandable to some degree since VR applications, and particularly VR applications for design, currently possess a large "wow" factor, we posit that there will be a shift toward more rigorous analysis of application of VR technology in the design process. This is especially true in light of the enormous potential VR technology, and particularly the low-cost current-generation VR technology, has to enhance the design process.

3.6 Conclusion

In the past few years, VR has come back into the public's awareness with the release of a new generation of VR products targeted at the general consumer. The low-cost, high-quality VR experiences these devices are capable of creating could prove a key enabler for VR to enter the engineering process in a more ubiquitous manner. When VR becomes a tool available to the average engineer, the tools discussed above as well as many more not yet imagined could become everyday realities. As shown above the applications span the entire development process from the initial early design phase through detailed design, product release and even into the rest of product life-cycle. This could significantly change the way engineering design work is done and allow new and innovative solutions to a wide variety of issues that today's engineers face. Wide-spread adoption of VR technology in engineering has the potential to be as pivotal a change to engineering as the introduction of the computer.

CHAPTER 4. CREATING A SHARED CONTEXT IN MULTI-USER VIRTUAL REALITY

4.1 Introduction

Before computers became commonplace in the engineering workplace, engineers working on a team project would often work around a shared drafting table [22]. In this shared space, designers could easily transition between collaborative and individual tasks. This work environment allowed designers to gather as needed to address questions, concerns, and new ideas, but still allowed parallel work on the project by each individual contributor. With the advent of the computer workstation, the work environment changed to a siloed, individualistic workflow, where collaboration activities require the designer to break-away from their design tools to have a discussion with their fellow teammates.

With the advent of Virtual Reality (VR) systems such as Head Mounted Displays (HMDs) and Cave Automatic Virtual Environments (CAVEs), researchers began studying how this technology could improve communication and collaboration [65]. However, due to cost [32, 66, 67] and limited availability of CAVEs and similar systems, the reality of a daily collaborative virtual environment (VE) has yet to materialize.

Over the past five years, the release of new low-cost, high-quality VR headsets is making large-scale multi-user applications, such as highly collaborative environment feasible, [241] even in a daily use scenario. However, there are still significant hurdles to be addressed before such systems become widespread.

One such hurdle is the issue of how to gather users who may be spread throughout the VE to the same virtual location in order to collaborate while avoiding cybersickness and disorientation. While there has been a large amount of work done in wayfinding and navigation in immersive environments, it typically focuses on active techniques where the user is in control [242]. However, when forming a shared viewpoint, it is advantageous to provide an automatic movement style in

order to reduce the amount of time required to form the shared view. This work studies the trade-offs between cybersickness and disorientation when moving users to a shared view in VR. We present the results of an experimental study comparing four different methods of creating a shared view in an immersive VE. The rest of this work will proceed as follows: Section 4.2 will review related research and motivation, Section 4.3 reviews the experimental method used, Section 4.4 reviews the results of the experiment, and finally, Section 4.5 presents a discussion of the results and implications for collaborative, immersive VEs.

4.2 Motivation

As discussed by Red et al. [22], previous to the general use of Computer-Aided Design (CAD) software the design environment of choice was the drafting table where collaboration and communication could flow freely as the need arose. Barbosa et al. [243] note that modern CAD systems have inherent architectural limitations which prevent true collaboration. In addition to barriers to collaboration from the software itself, designers face additional challenges from globally distributed design teams [2] and Chen et al. [244] note that standard collaborative software is insufficient to the task due to the complexities of collaborative design activities.

Various approaches for bridging this collaboration gap have been tried. Red et al. [245] created a system to allow multiple users to natively access and modify CAD designs simultaneously. Nysetvold et al. [246] created a module to allow view sharing and presense awareness in CAD packages. Since at least 1997 researchers have also studied integrating CAD systems into VR environments [179]. Immersive VR holds significant promise as a collaborative design environment. Past research has shown immersive VR tools allow easier comprehension of 3D geometry, they can support a collaborative multi-user environment [8], and can be distributed [51]. Berg et al. [247] have even shown advantages to holding design reviews in VR. However, with all its advantages, traditional VR technology falls short of offering an ideal solution due, in part, to high cost and resource demands. This has caused severely limited access and hence limited solutions to the issues of collaboration. Additionally, since traditional VR hardware can only support one or two unique viewpoints at a time without expensive duplication, resource cost considerations prevent its use as a design system when users are working individually. This means it has a similar barrier to

collaboration as standard CAD because a user must still leave his design environment in order to enter a collaborative design environment.

New consumer level VR hardware is beginning to address these issues with costs that are just a fraction of traditional systems. For example a low cost traditional setup might currently cost $100,000, support one or two unique view points, and a group of eight people at a time (not to mention the space requirements of the setup). For that same price, enough consumer-level hardware could be purchased to support up to 50 designers in an immersive collaborative design tool each with their own unique viewpoint. Not only would this allow each designer to have more regular access to VR tools, it also supports a collaborative environment in which each designer can work independently (since each has their own viewpoint) or collaboratively, much like working at a drafting table. This unprecedented availability makes new applications such as a networked, immersive, collaborative design environments economically feasible. However, there are still additional barriers to creating an effective VR collaboration environment. Such barriers include cybersickness [36], reproducing non-verbal communication [248], specialized interaction controls [249], and creating a common frame of reference (COFOR) [250].

In order to effectively collaborate, the collaborators must first form a COFOR which is a shared mental representation of the geometry, task, problem, and current situation. Methods to form this COFOR generally include both verbal and non-verbal cues, but for non-verbal cues to be effective, collaborators must be able to see the same thing. In collaborative software, the ability to see the same things is often accomplished through view sharing. In fact, Valin et al. [57] identified the ability to share a common view as a requirement for a collaborative environment. However, in immersive collaborative VR environments creating a shared view presents a problem. Users who are working individually must all converge on the shared view. While this is a relatively simple problem computationally, it is complicated by two aspects of the user experience. The first is that moving the user tends to produce cybersickness in the user [251]. Symptoms of cybersickness range from mild to severe and, in extreme cases, the user can become so physically ill they must discontinue use of the environment. The second issue is that moving the user can break the user's locational frame of reference causing disorientation and confusion about where they are in the environment [252]. This locational frame of reference must typically be reestablished before users can effectively identify items in the shared view and hence begin creation of a COFOR.

In many current consumer VR experiences teleportation is the predominant movement style. While the Oculus Best Practices guide [253] suggests teleportation as a movement style for its low likely-hood of causing cybersickness, it also notes that this movement style can be particularly disorienting to the user as shown by Bowman et al. [252]. Additionally, the Oculus Best Practices guide suggests that a VE should support multiple movement styles and allow users to pick their preference. This work presents an experimental study that was conducted to understand the trade-offs between cybersickness and disorientation from four different styles of shared view creation. The transition styles considered are: Teleport, Fade, Fly, and Manual. In this study we also seek to understand user's transition style preferences as it relates to shared view creation in VR.

This results of this study supports the findings of [252] that teleportation has the potential to produce high amounts of disorientation in users when used to create a shared view. Furthermore, we hypothesized that using a fly-around transition style (in which users are automatically moved along a path from their current location to the shared view location) to create a shared view would produce much less disorientation, potentially at the cost of increased cybersickness as discussed by Elvezio et al. [254] We hypothesized that under manual movement (in which participants use a gamepad to move themselves) participants should experience the least amount of disorientation, but that the time to move to the shared view could take excessively long. Finally, we also hypothesized that the fade-in/fade-out style will perform similarly to teleport in both cybersickness and disorientation.

4.3 Methodology

4.3.1 Overview

For this study, a VE (shown in Figure 4.1) was created which consists of a white background with a faint horizon line and a false color model of a Lamborgini Reventon scaled to approximately 2x normal size relative to the user. This environment also supported moving the user to a designated location in the VE via all four of the methods described in Section 4.2, namely: Teleport, Fade, Fly, and Manual. For the study, 42 unique parts of the car were chosen and participants were asked to identify in which quadrant of the car (shown in Figure 4.2) each part is located

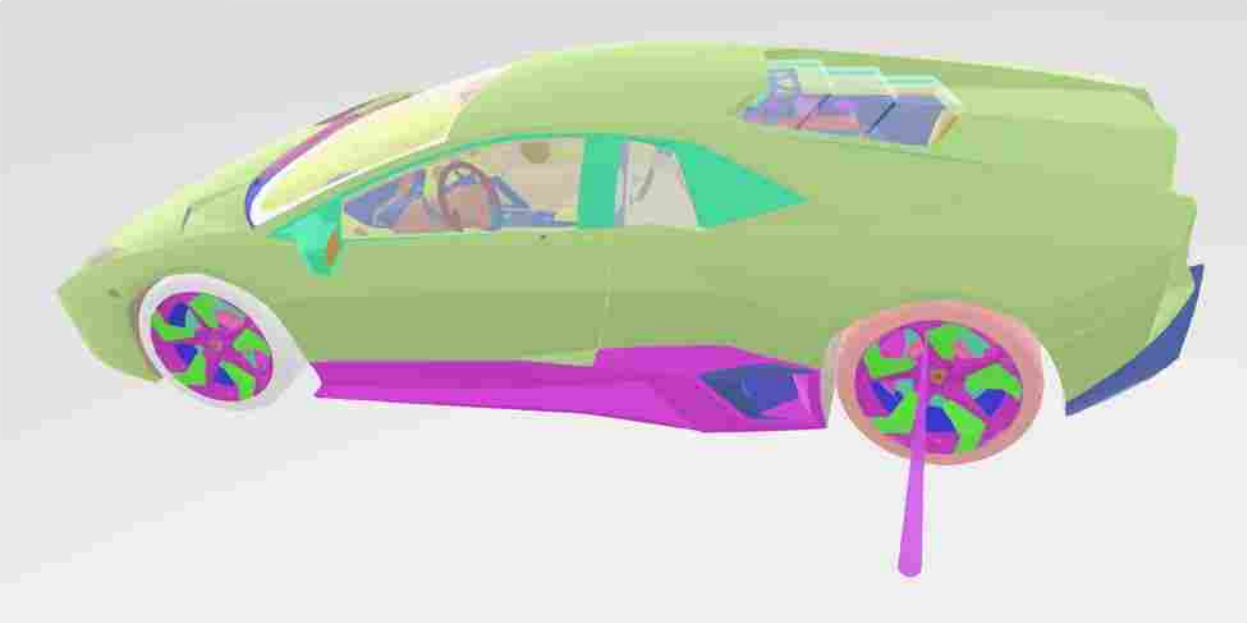


Figure 4.1: View of the VE in mid fly transition. The driver's rear tire is the part for which to identify the quadrant.

after being moved to a viewing location via one of the four transition styles. In order to assess the amount of disorientation caused by the view creation styles, the time to identify the quadrant, the participant's stated confidence, and the correctness of their response were recorded. To asses the amount of cybersickness induced by each transition style each participant identified eight locations on the car (randomly-chosen from the total 42) using the same view creation style which we defined as a block. After completing a block of eight identifications, participants were asked to verbally complete the Simulator Sickness Questionaire (SSQ). Each participant was asked to complete five identification blocks, one block of eight for each transition style plus one more block of 10 where the transition styles were randomly mixed together. After completing the experiment, each participant was asked to complete a survey about their experience and preference.

4.3.2 Participants

All participants were recruited from Brigham Young University for this study. Participants ranged from pre-college to recent graduates with the majority being in their first or second year of college. All participants majored or intended to major in a Science, Technology, Engineering, or Math (STEM) related field. Of the 45 datasets collected, time data for two participants was

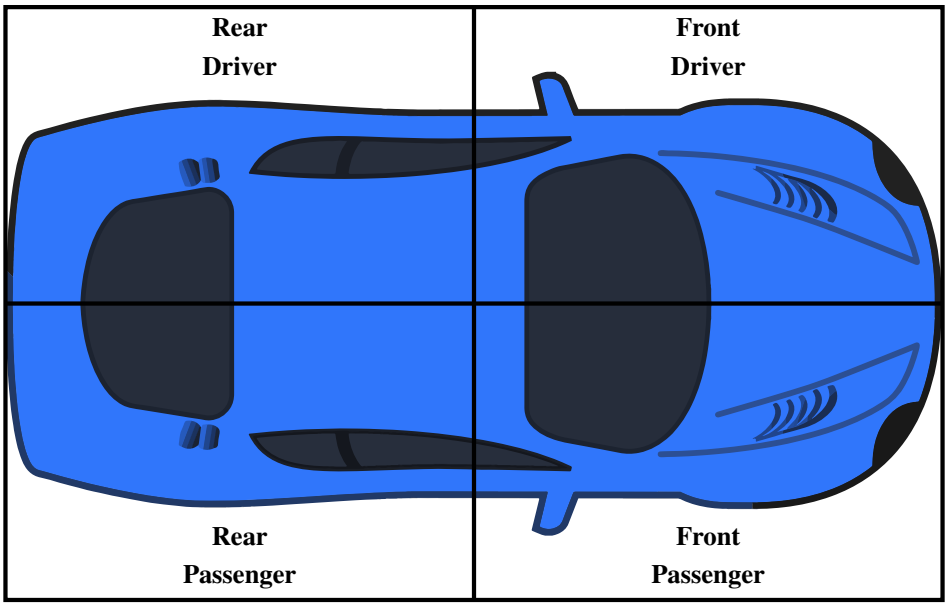


Figure 4.2: The quadrants of the car used to identify part location.

lost due to technical difficulties. Their response data has been excluded where appropriate. Of the 45 participants, 33 had 0–5 hours of experience with VR, eight had 5–10 hours of experience, two had 10-30 hours, and two had 30 - 100 hours of experience with VR. No compensation was provided for participation. Although participants were allowed to terminate their participation early should the cybersickness symptoms become uncomfortable, each participant completed the entire experiment.

4.3.3 Aparatus

An Oculus Rift CV1 HMD was used as the VR display and was driven by a personal computer (Intel Xeon W3520 2.67GHz, 12GB RAM, and an NVIDIA Geforce GTX 970). Players used an XBox One controller which was linked wirelessly to the computer to control the manual movements. During the tasks involving automatic transition styles the manual controls were inoperative. All development of the VE was done in Unity3D version 5.4.2. All time data was collected using built-in Unity functions and was measured from the start of the view creation until the time the participant identified the quadrant in which the part was located after which the proctor stopped the timer.

4.3.4 Tutorial

Prior to beginning the experimental task, participants were given a tutorial on the VR environment and the task they would perform. For the tutorial, an environment similar to experimental environment was created except that the model of the Lamborgini was replaced with a false color model of a batmobile in order to avoid pre-familiarization with the experimental model. During the tutorial participants were instructed on the use of the manual controls and then allowed to practice with them until they felt comfortable with the controls. This lasted up to several mintues. After users were familiar with the manual controls, they were asked to identify the quadrant of the car in which each of five highlighted parts was located. In the event that users made a mistake identifying the quadrant, they were corrected in order to avoid misunderstanding about the location and name of each quadrant.

4.3.5 Experimental Task

For this study 42 unique parts on the car were chosen as well as a corresponding shared view location and orientation for each part. Parts were chosen that either had four duplicates in the model (e.g. the tires), two duplicates in the model (e.g. the side view mirrors), or no duplicates in the model (e.g. the hood ornament). From among these 42 parts, 10 were randomly chosen, ordered, and assigned transition styles to form the final fifth block of mixed creation styles. This block was the same for each participant. For each participant, the remaining 32 parts were then randomly divided into four blocks of eight parts such that each participant identified each part only once. Each block of eight was randomly assigned a view creation style that would be used for the entire block such that the view creation styles were presented to each participant in a random order.

In order to ensure that participants knew which part they were to identify, the location of (and in the case of the manual transition style to help them find it) the part for identification was highlighted by blinking it red and by the addition of an approximately two meter long translucent cone which pointed to the part.

An example of the experimental task would be as follows. The proctor would press a key to initiate the transition. The program would first change which part was highlighted and the location of the cone. It then starts a timer and begins the transition to the shared view location. Once the

transition was complete the participant was asked to identify the quadrant of the car the highlighted part was in. Participants were allowed to use the head tracking capabilities of the HMD to look around them, if needed, to regain their bearings. Once the participant identified the quadrant, the proctor would press a second button to log the time. The participant was then asked their confidence (on a scale of 0 – 100%) regarding the location of the part. The proctor would record the quadrant and confidence responses and then begin the next transition.

The manual transition style differed slightly. The proctor would press a key to start a transition which would change the highlighted part and location of the cone and start the timer. The participant was also verbally notified that a new transition had occurred. However, instead of being moved automatically the participant would use the manual controls described in Section 4.3.6 to move themselves around the environment to find the highlighted object. Time was stopped when they identified the quadrant and confidence data was collected as described.

After a participant had completed eight transitions of a given type they would complete a SSQ survey. The SSQ was administered verbally by the proctor to minimize the number of times the participant had to remove and put-on the VR headset. After a participant had completed the four randomized blocks of a single transition styles, they completed the fifth and final block of mixed transition styles.

4.3.6 View Creation Styles

As noted previously four different view creation styles were tested in this study which were: Teleport, Fade, Fly, and Manual. Teleport consisted of immediately placing the user at the shared view location and orientation. For Fade one second was taken to fade the user's view to black, then the user was teleported to the new location and orientation. Finally one more second was taken to to fade the user's view back in from black. This resulted in a total transition time of two seconds. For the Fly transition, a path was dynamically calculated that would take the user out from their current location to a view of the entire car and into the final location. For more information on how this path was calculated see Section 4.3.6. The entire fly transition was completed in three seconds and movement speed was scaled according to the total path length. Additionally the movement speed was smoothly ramped from 0 to the peak speed for that movement and back down to 0. An example velocity plot can be seen in Figure 4.3. As mentioned previously the Manual transition

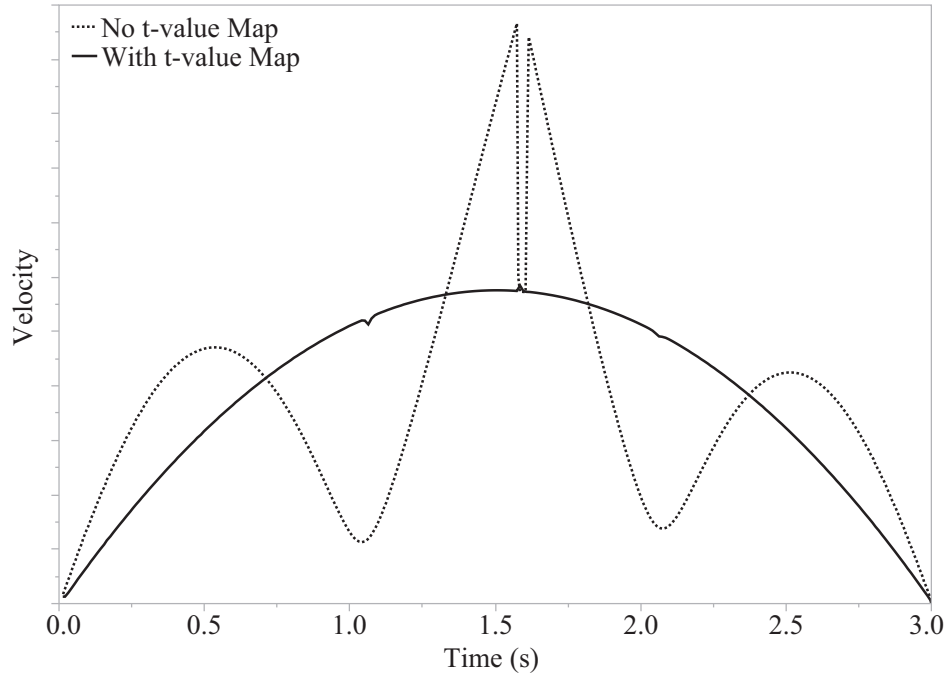


Figure 4.3: A comparison between movement velocities over the transition time when using and not using a t-value to curve-length map. Note the significant velocity drop on the No t-value Map graph around 1.5s. This small drop is when the path travels along the sphere arc instead of a Bezier curve.

style consisted of creating a new shared view (which changed the highlighted object) and then allowing the participant to find it on their own using the manual controls described in Section 4.3.6. No additional hints were given as to the location (such as a minimap as might be common in a video game) since this would confound the quadrant identification accuracy from the transition style itself.

Manual Controls

Since this environment was similar to some video games the manual control layout from those games were mimicked. However, certain modifications needed to be made since most games lock players to a floor in the environment, but the experimental environment allowed a user to move in 3 dimensions in order to allow participants to view the car from any angle. This resulted in additional degrees of freedom which participants needed to control. Hence, in addition to the standard left/right, forward/backward, pitch and yaw controls found commonly in games, participants were also given control over up/down movement. For the manual controls, all of the movement axes

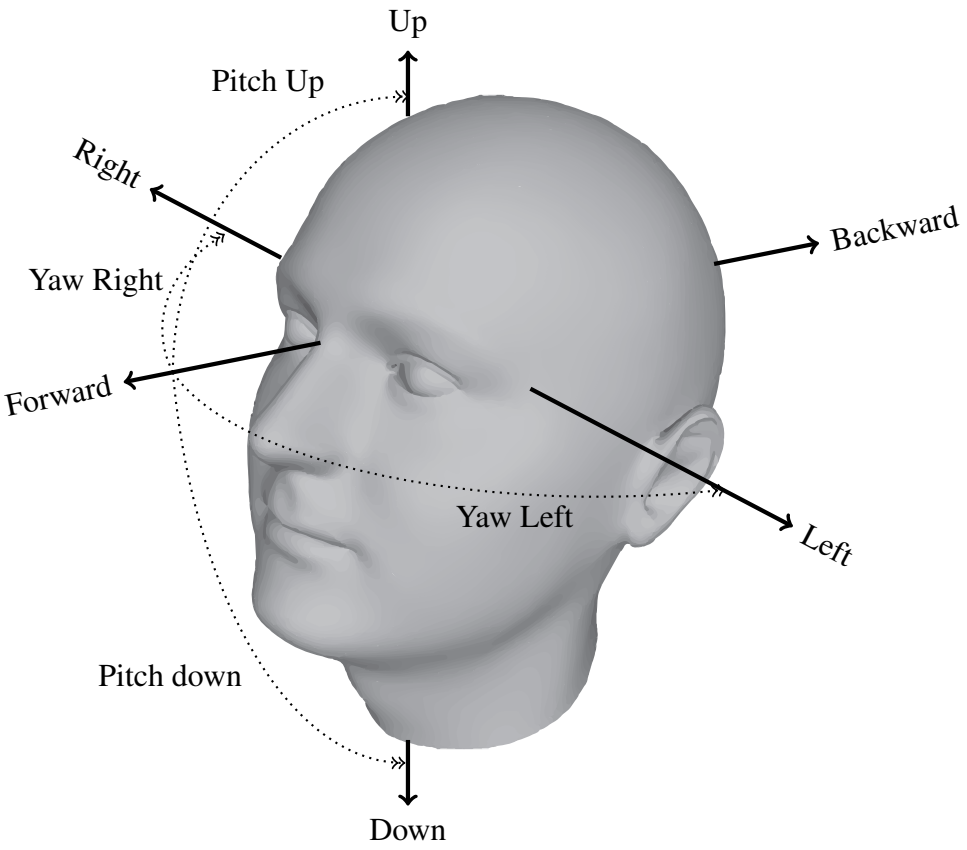


Figure 4.4: The movement directions for the manual movement style are relative to the user's gaze as shown here.

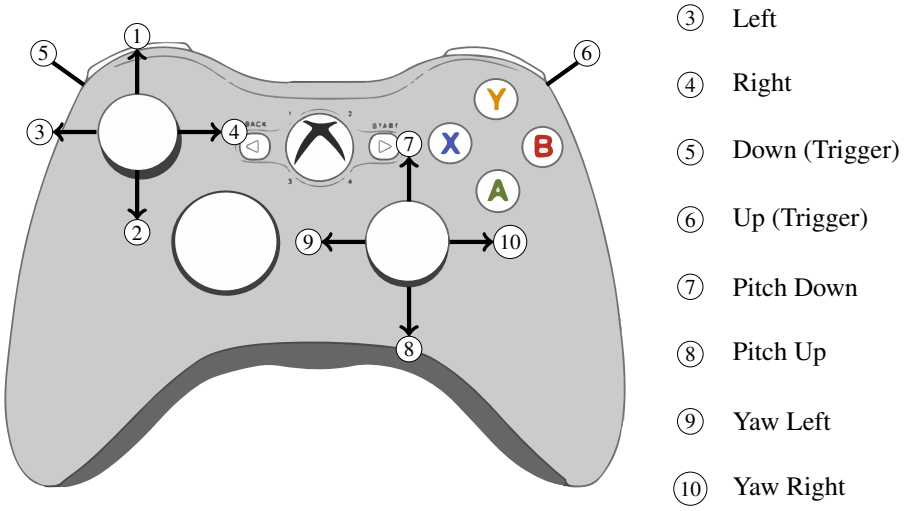


Figure 4.5: The gamepad controls for the manual movement style.

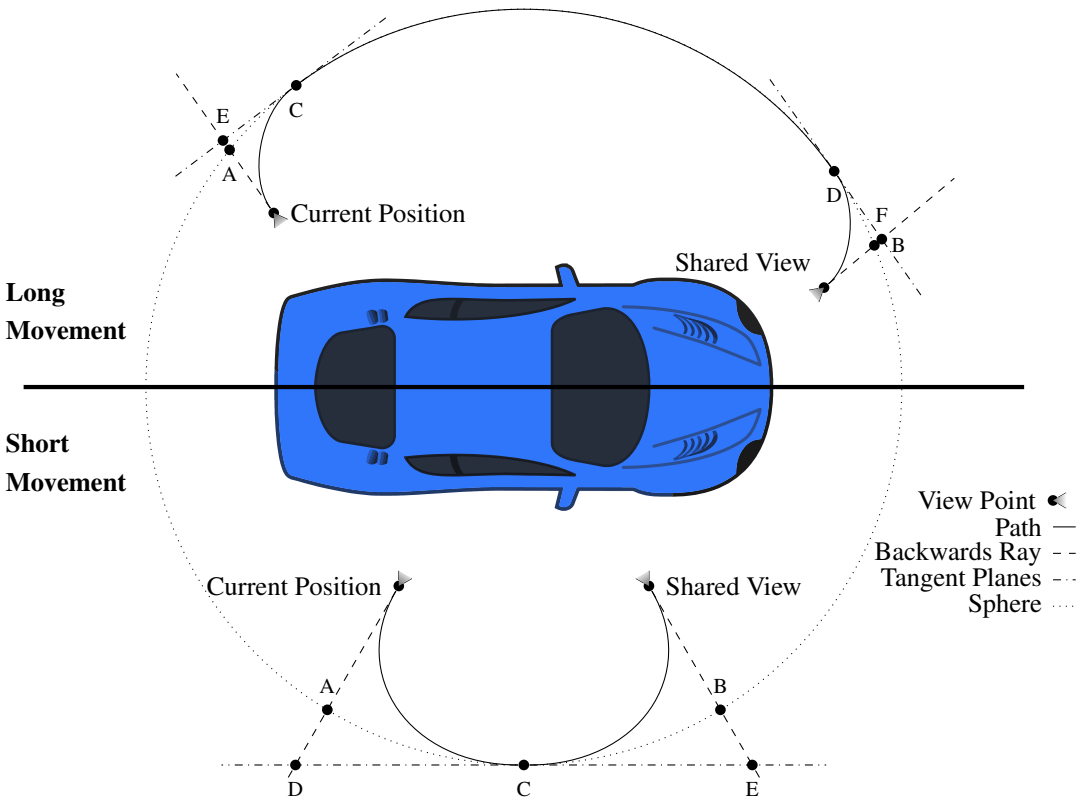


Figure 4.6: Shown are the points, rays, planes, and sphere used to calculate the path for the Fly movement. The top half shows a longer fly where a portion of the path is an arc along the sphere. The bottom half shows a shorter fly movement that doesn't need to fly along the sphere. For the longer movement style, points Current Position, E, C as well as points D, F, Shared View form the control points of the Bezier Curves. For the shorter movement, the controls points for the two bezier curves are Current Position, D, C and C, E, Shared View.

were defined relative to the user's gaze as shown in Figure 4.4. Head position and orientation data was taken from the head tracking capabilities of the Oculus Rift. The inputs on the controller corresponding to each movement axis are shown in Figure 4.5. Note that each of the inputs used on the controller reports the degree to which that input is activated instead of a more typical boolean on/off. This functionality was used to create controls that would move the participant faster the further the control was displaced.

Creating a Path for Fly

In order to avoid jarring direction changes, two degree-two Bezier curves and potentially an arc segement (on longer distance movements) were calculated that would result in a tangentially

continuous path. Figure 4.6 shows an example of each path style. The paths were calculated as follows. First, a sphere of radius 10m was defined, centered at the car. Then the ray opposite of the participant's current look direction was intersected with the sphere to create point A. Next the backwards ray of the shared-view look-direction was intersected with the sphere to create point B. The arc distance along the sphere between points A and B was then calculated. If the arc length was less than 20m, the middle point of the arc between points A and B would be chosen as point C. The plane tangent to the sphere at point C would then be found and the same backwards rays used to define points A and B would be intersected with the plane to find points D and E. The path would then consist of two Bezier curves defined by the control points {Current Location, D, C} and {C, E, Shared View Location}. These two curves can be shown to be tangentially continuous since points C, D, and E all lie in a straight line. If the arc distance between points A and B was greater than 20m, points C and D are defined as points on the arc 10m from points A and B respectively. The planes tangent to the sphere at points C and D are then calculated and the backward rays used to define points A and B are intersected with the planes from C and D to find points E and F respectively. The path would then consist of the two Bezier curves defined by the control points {Current Location, E, C} and {D, F, Shared View Location} as well as the arc along the sphere from C to D which again can be shown to be tangentially continuous since the lines EC and DF are tangent to the sphere.

Since the parametric t-value of a Bezier curve does not sweep out the curve at a constant speed, a map of 10,000 t values and corresponding curve-length values was created for each Bezier curve. Thus the program could then know the total path length and hence find the desired distance along the path the participant should be at any given time. This desired distance along the path could then be mapped to an interpolated t-value on the appropriate Bezier curve to create a smooth velocity along the path. Figure 4.3 shows a comparison of movement velocity over time with and without the t-value map.

The path calculated with this algorithm will start at the user's current location and move them backwards along the curve until reaching a point 10m from the center of the car, then along the sphere if needed, and then down into the shared view location along a path tangential to the shared view look direction. This path will move the participant smoothly from their current location to shared view location while allowing them a full view of the car for at least a portion of the

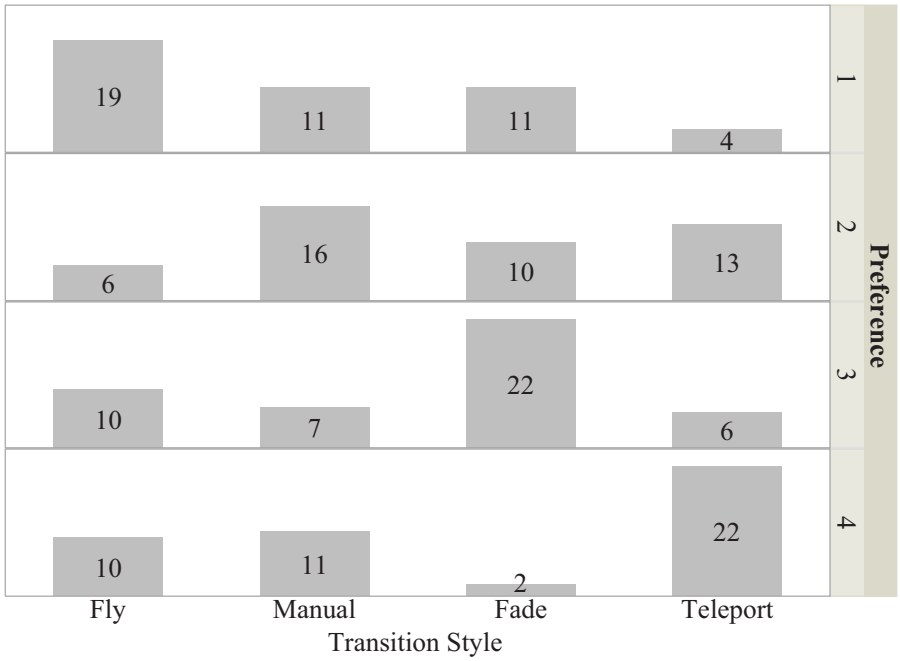


Figure 4.7: Frequency count of participant's transition style preferences from 1 (most preferred) to 4 (least preferred).

path. In order to create a smooth transition from the participant's current orientation to the shared view orientation, the orientation on the path is implemented as a linear interpolation of the start and end orientations with the amount of interpolation corresponding to the distance along the path.

4.4 Results

As mentioned, the Oculus Best Practices document [253] suggests that VEs implement multiple movement styles and allow users to choose their preference. Figure 4.7 shows the breakdown of participant's transition style preferences for creating a shared view. Fly was the most commonly preferred method with aproximately 43% of participants choosing it as their preferred transition style. Teleport was the least commonly preferred transition style with only 9% choosing it as their preferred style. Both Manual and Fade were chosen as the preferred style by approximately 24% of participants. Also from Figure 4.7 we can see that Manual was the most common choice for second preference, Fade for third preference, and Fly as least preferred.

When asked why participants chose Fly as their most preferred style, reasons centered on the speed of the transition coupled with increased locational awareness. Users stated: "It gets me

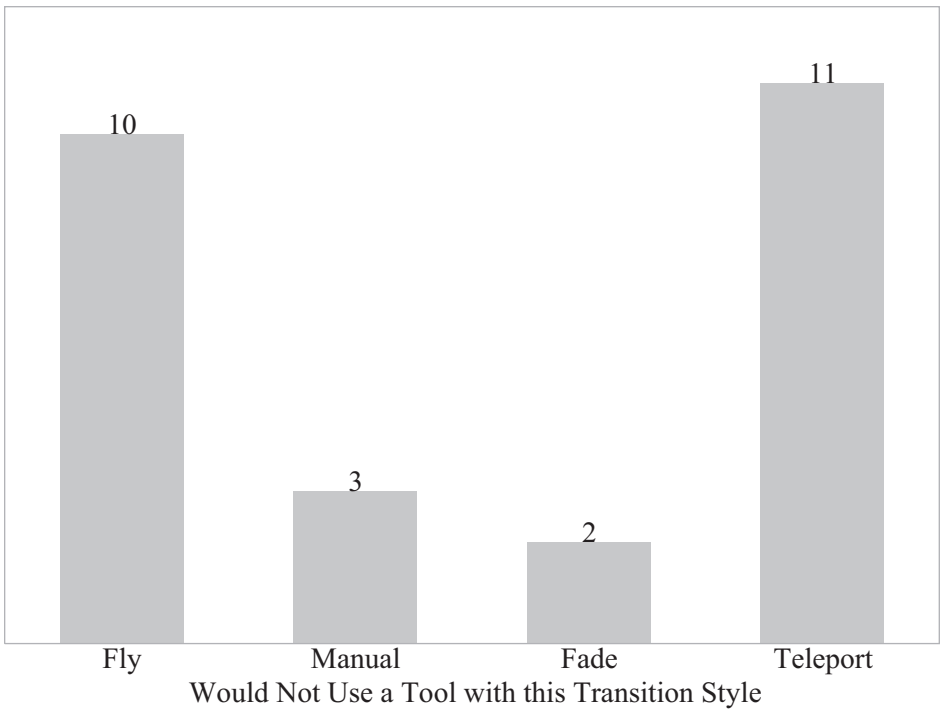


Figure 4.8: Number participants who reported they would not use a tool if the listed transition style was the only available style.

directly to the location [of the shared view] while 'zooming' out so I can view the car from a distance" and "It's faster than manual, but still allows me to see where I am moving to from where I was."

Participants who preferred Manual generally gave reasons focused on the increased control. Comments included: "I preferred to find the angle myself", "I always felt in control of where I was", and "It allowed me to regain my bearings if I got lost."

Reasons for preferring Fade generally focused on the comfort of the transition. For instance: "Comfortable transition ... easy on the eyes", "It was a gentle transition", and "Less abrupt/jarring."

Finally, reduced cybersickness was the most common reason for preferring Teleport. Responses included: "Least amount of cyber sickness" and "It hardly made me sick."

Figure 4.8 shows the number of participants who reported they would not use a collaborative VE if the listed transition style was the only available style. Notably, approximately a quarter of the participants would not use a tool that only provided a Fly transition style even though fly

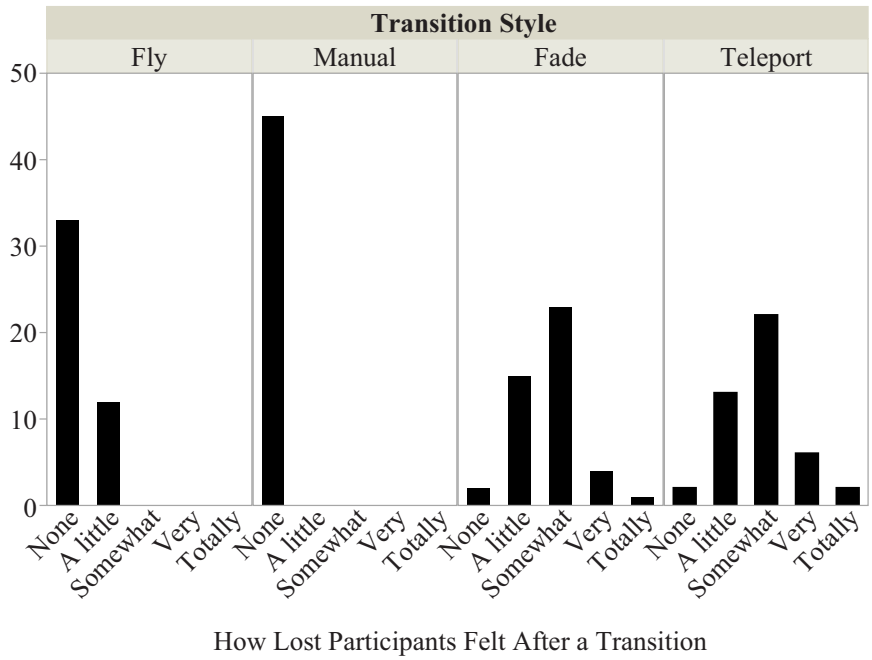


Figure 4.9: Frequency count of participant's rating on how lost they felt on average after a transition of the listed style (rated on a five point Likert scale).

was the most popular choice for a participant's preferred transition style. This finding underscores the need for VEs to include multiple transition styles.

Figures 4.9, 4.10, and 4.11 show feedback regarding the various transition styles that was collected in the post-experiment survey. From Figures 4.9 and 4.10 we see that participants felt less lost after a transition and less disoriented during a transition with both Fly and Manual than either Fade or Teleport as was hypothesized. Figure 4.11 shows how participants rated the physical discomfort (cybersickness) of the Fly, Manual, and Fade styles compared to the Teleport style. As expected Fade has a normal distribution with a mean value of slightly less discomfort than Teleport. In contrast, both Fly and Manual appear to show bimodal distributions for discomfort compared to Teleport. The primary mode in both distributions centers on more discomfort than Teleport as expected, however, the secondary mode centers on much less discomfort than Teleport. This second mode could be explained by participants who found the sudden jump to a new location uncomfortable and hence found Fly and Manual less uncomfortable. Support for this theory can be found from the fact that 60% of participants who rated Fly as much less uncomfortable than Teleport also said they disliked Teleport enough to not use the tool if it were the only available transition style. Similar results are found for participants who rated Manual as much less uncom-

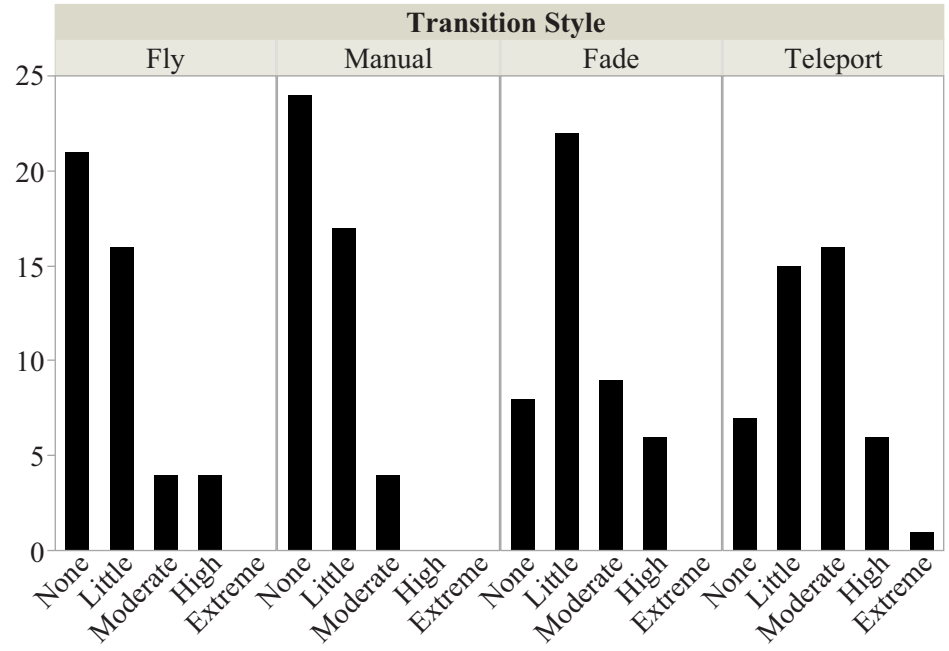


Figure 4.10: Frequency count of participant's rating of how disoriented they felt on average during a transition of the listed style (rated on a five point Likert scale).

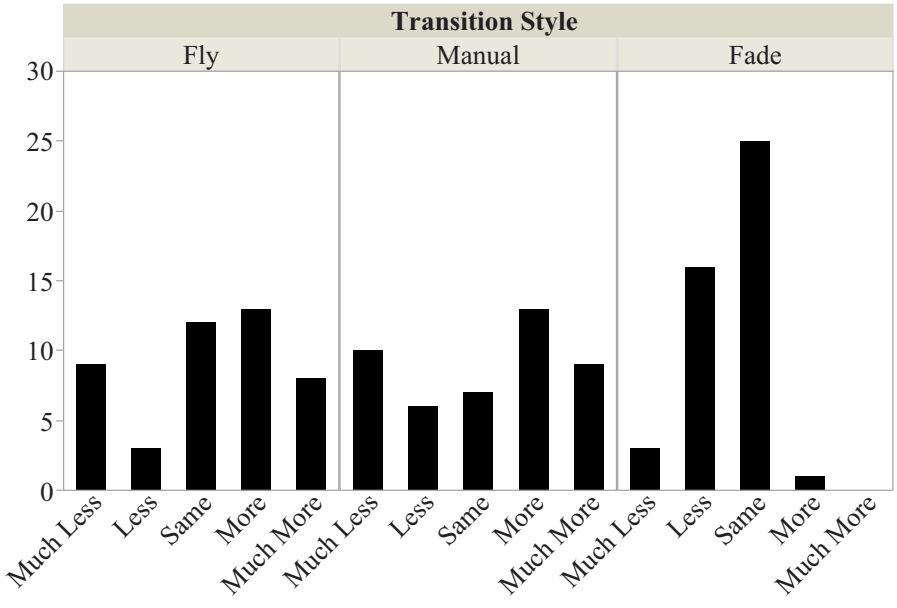


Figure 4.11: Frequency count of participant's rating of how disoriented they felt on average during a transition of the listed style (rated on a five point Likert scale).

Table 4.1: Number of Incorrect Responses by Transition Style

Fly	Manual	Fade	Teleport
4	3	15	20

Table 4.2: Mean and Std. Dev. of Confidence(%) regarding Part Quadrant by Transition Style

	Fly	Manual	Fade	Teleport
Mean	99.0	99.5	95.3	95.8
Std. Dev.	4.12	2.00	11.62	12.08

fortable than Teleport. Additionally, these two sub-populations frequently described Teleport as "jarring visually", "most strain on my eyes", and "very sudden/abrupt".

Table 4.1 shows the total number of times participants incorrectly identified the quadrant location of a part across the different transition styles. As seen, Fade and Teleport have a 3x to 7x increase in the number of incorrect answers compared with Fly and Manual. Figure 4.12 shows participants' stated confidence for quadrants they identified incorrectly grouped by transition style. Interestingly, some participants had 100% confidence in an incorrect answer. Some of these may be caused by communication errors rather than gaps in understanding. For instance, participants would occasionally identify a part as being on the passenger side when they knew it was on the driver's side and then quickly correct their communication error. It can be reasonably assumed that there were additional communication errors that were not corrected. If it were possible to remove these additional communication errors from this data set, Fade and Teleport would likely show even larger increases in error rate over Fly and Manual.

Table 4.2 shows participants' mean confidence that a part is actually located in the quadrant they believe it is in as well as the standard deviation based on the transition style. Table 4.3 shows the p-values of paired t-tests of the confidence data by transition style. As can be seen from the p-values and means, Fade and Teleport are not significantly different from each other. However, both Fly and Manual provide a statistically significant increase in confidence of approximately 4%.

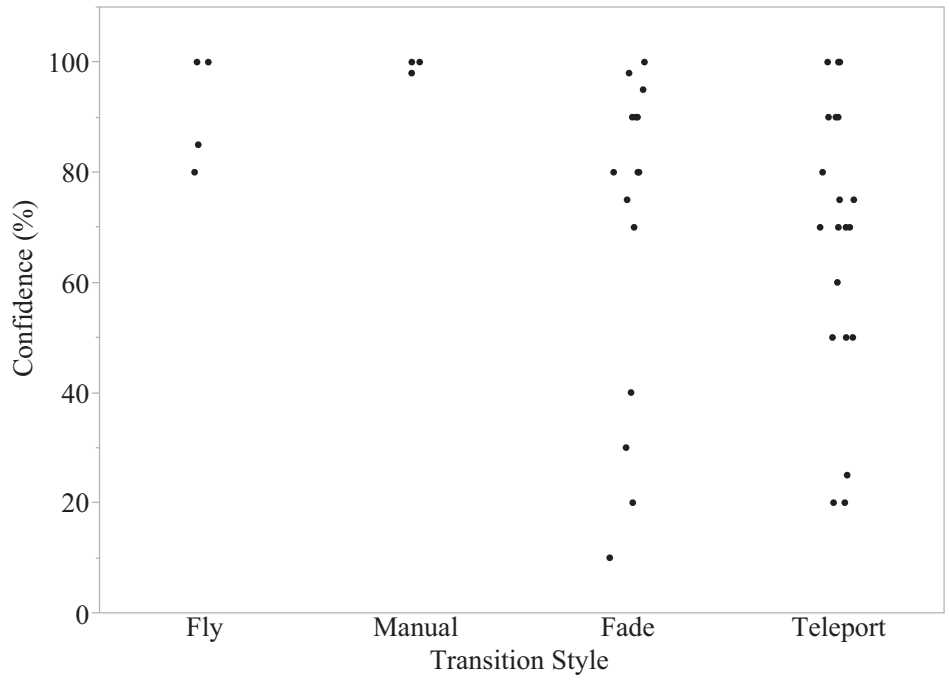


Figure 4.12: Participant's stated confidence for quadrants they identified incorrectly grouped by transition style.

Table 4.3: P-Values from Paired t-Test Comparison of Confidence by Transition Style. Bold values indicate significance at the 95% confidence level.

	Fly	Manual	Fade	Teleport
Fly	-	**0.037**	**<.001**	**<.001**
Manual	-	-	**<.001**	**<.001**
Fade	-	-	-	.400

Table 4.4 shows the mean time to identify the quadrant location of a part by transition styles as well as standard deviations. Note that these times include the time required for the transition (3s for Fly, 2s for Fade, variable for Manual). Fade on average took approximately two seconds longer than Teleport which was the exact amount of time used for the transition. Fly, on the other hand, took almost the same amount of time as Teleport including the three second transition time. In addition, the standard deviation was noticeably lower. Table 4.4 also shows that the mean time for Manual was significantly longer than the other three styles. Some of this time difference is due to the fact that for certain parts manual was more of a test of how quickly participants could find the

Table 4.4: Mean and Std. Dev. of Time(s) Required to Identify Part Quadrant by Transition Style

	Fly	Manual	Fade	Teleport
Mean	7.73	17.03	8.91	7.68
Std. Dev.	3.39	15.31	6.68	5.19

Table 4.5: P-Values from Paired t-Test Comparison on the Time by Transition Style. Bold values indicate significance at the 95% confidence level.

	Fly	Manual	Fade	Teleport
Fly	-	$<$**.001**	$<$**.001**	.096
Manual	-	-	$<$**.001**	$<$**.001**
Fade	-	-	-	.025

part, and not necessarily identify it (Figure 4.13). In some cases, such as parts 5 and 23, the time for Manual was on approximately the same as the other three styles. However, in other cases, such as parts 15 and 40, the part and pointing cone were partially or fully occluded from many angles requiring significant amounts of time to locate them. Table 4.5 shows the p-values from paired t-tests of the time across transition style. This analysis indicates that all the pairs are statistically different except for Fly and Teleport, and Fade and Teleport. Hence from the trends in Tables 4.4 and 4.5 we conclude that the time spent on the Fly transition was useful and helped improve user's locational awareness compared to the standard Teleport transition style.

Table 4.6 presents the mean, standard deviation, minimum, and maximum SSQ scores from the four different transition styles calculated via three different scoring systems. The original questionnaire and scoring system was presented by [255] and was developed for navy pilots using flight simulators. According to the data presented in the same article, our means fall in the 65th–70th percentiles and our maximums fall in 98th–99th percentiles. However as noted by [257] their test subject population is unlikely to be representative of the general population for reasons such as physical condition, experience with highly dynamic environments, potentially lower occurrences of motion sickness, and differences in the experience flight simulators provide compared to VR hardware. These population differences could explain our above average means. When compar-

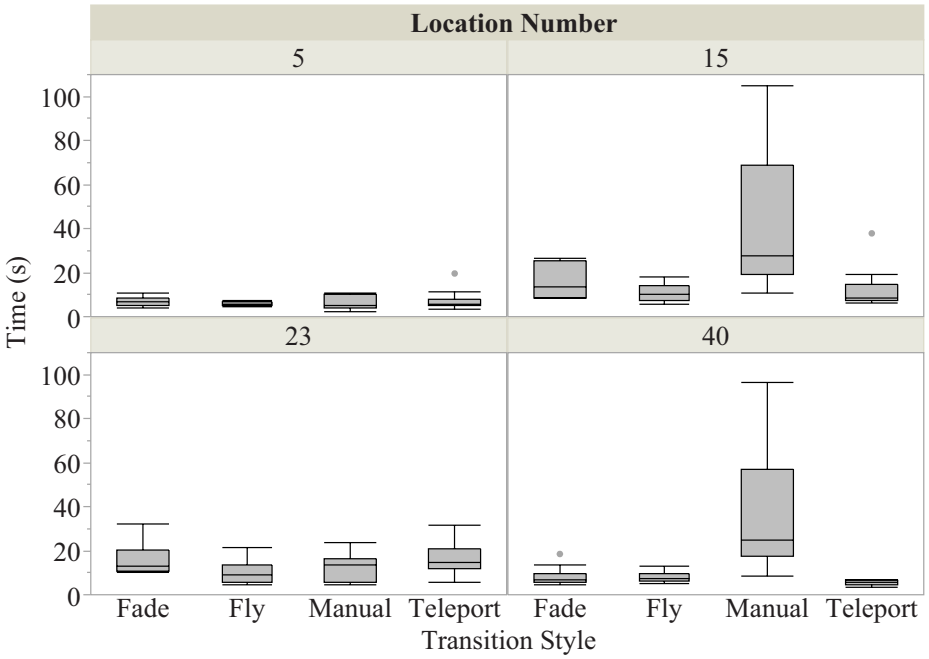


Figure 4.13: Examples of the amount of time taken to identify the quadrant of a given part broken down by transition style.

Table 4.6: Mean, Std. Dev., Min, and Max SSQ scores calculated with the weights originally presented by Kennedy et al. [255], alternate weights presented by Bourchard et al. [256], and raw scores as discussed by Bourchard et al. [257].

	Mean	Std. Dev.	Min.	Max.
Original Scoring (weighted)				
Fly	11.72	14.61	0	59.84
Manual	14.88	19.03	0	108.46
Fade	8.73	15.32	0	86.02
Teleport	8.81	13.15	0	67.32
Alternate Scoring (weighted)				
Fly	1.47	1.76	0	7.23
Manual	1.86	2.24	0	12.78
Fade	1.10	1.83	0	10.62
Teleport	1.15	1.55	0	7.90
Raw Scoring (unweighted)				
Fly	2.29	2.74	0	11
Manual	2.87	3.49	0	20
Fade	1.71	2.87	0	17
Teleport	1.76	2.45	0	13

Table 4.7: P-Values from Paired t-Test Comparison of SSQ scores by Transition Style. Bold values indicate significance at the 95% confidence level.

	Fly	Manual	Fade	Teleport
Original Scoring (weighted)				
Fly	-	.136	.122	.092
Manual	-	-	**<.001**	**.010**
Fade	-	-	-	.964
Alternate Scoring (weighted)				
Fly	-	.119	.117	.097
Manual	-	-	**<.001**	**.011**
Fade	-	-	-	.839
Raw Scoring (unweighted)				
Fly	-	.134	.113	.082
Manual	-	-	**<.001**	**.013**
Fade	-	-	-	.900

ing the original scoring results and raw scoring results to the data presented by [257] the means found in this study are approximately one third of those seen by Bouchard. However, Bouchard's population consisted of patients with clincal phobias using VEs to treat their phobia and hence their population is also not representative of the general population. The alternate scoring method presented by [256] was an attempt to provide a method that does not double count any symptom on the questionnaire with weights based on a more general population, but little data is available using this scoring method. Hence, we present our data for potential comparisons with others.

When a paired t-tests were performed on the SSQ scores across the transition styles (see Table 4.7), statistically significant differences were found for Manual vs Teleport, and Manual vs Fade. Additionally, the means presented in Table 4.6 as well as the data shown in Figure 4.11 suggest that there may be more differences in the amount of cybersickness caused by the different transition styles which could be revealed with larger sample sizes.

4.5 Discussion and Recommendations

From the data presented in Section 4.4, the Teleport transition style produced high amounts of disorientation as hypothesized. This corroborates the findings from [252]. Interestingly, this data

also shows that, for at least a subset of the population, Teleport is a rather uncomfortable transition style. Fade provides an alternative to Teleport that performs similarly but requires slightly longer reorientation times with slightly reduced discomfort levels due to it being a less sudden transition.

Fly performed much better than either Fade or Teleport in locational awareness, confidence, and accuracy with total reorientation times similar to Teleport. However, Fly gains these advantages at the potential cost of increased discomfort—especially for more sensitive users. Interestingly, this is a trade-off that many users seem to prefer. Also, feedback from users suggests that the discomfort of fly might be reduced by slowing down the travel speed (increasing the reorientation time). It was also observed during the study that many participants were uncomfortable (and disoriented) when the Fly path took them through a solid object. It was also observed that during manual movements participants typically moved around solid objects and through windows if possible to avoid traveling through a solid object. Hence, improving the Fly path to not travel through solid objects may further reduce discomfort.

Manual performed similarly or better than Fly in locational awareness, confidence, and accuracy, but at the cost of significantly increased time and potentially additional cybersickness. However, much of the time increase was due to the fact that participants had to find parts that were typically not immediately visible and often mostly obscured. This could potentially be mitigated by better navigation aides such as bird's eye views, and navigation arrows as discussed by [258]. Additionally, many participants commented that they would like to be able to control the roll since they would often get turned off axis from vertical after a series of manual movements. Participants were also generally divided on the intuitiveness of the control scheme. Therefore providing both a control for roll and a more intuitive control set could reduce both the discomfort of the manual style and the time required to move to the shared view location.

Finally, this study underscores the importance of providing multiple transition options as recommended in the Oculus Best Practices document [253]. As seen in the results, every transition style had at least a few people who would not use a tool which supported only one of the transition styles. Thus, in order to provide a VE that the widest audience can use to collaborate it is very important that multiple transition styles be provided so users can choose their preferred trade-off between speed, location awareness, and discomfort.

The most critical limitation of this study is that participants experienced each transition in quick succession as opposed to a true collaborative environment where users would experience transitions much less frequently since discussions would take place after a transition occurred and users would likely spend time working individually as well. This reduced transition occurrence could potentially change user's style preferences and impact the SSQ scores observed.

Additional limitations are found in the sample population and sample size. Participants in this survey volunteered based on their interest in VR technology and were typically pursing a technical undergraduate degree. Hence, participants were predisposed in favor of VR and may have been willing to endure higher rates of cybersickness than the general population.

As noted previously both the Fly and Manual transition methods could be improved. Improvements to these styles could make them more preferable. This sentiment was expressed by a few participants who mentioned that manual could potentially be their preferred style with some improvements.

Additionally, some participants suggested a combination of Fade and Manual or Fly. This may provide the most of the benefits of each method while reducing their downsides. This combination method would use Fade to move the user to a position from which they could see the shared view location in relation to the overall design and then Fly or allow users to move manually to shared view location on their own from that point. This method could potentially be used to reduce the number of solid objects a user has to move through to get to the shared view location as well as eliminate the time required for a user to find the shared view with the Manual transition style.

Additional studies should also be conducted that study the transition styles presented in a collaborative task to evaluate changes to user preferences in these settings.

4.6 Conclusion

This work conducted a study which exposed participants to various styles of transitioning to a shared view for VR. After a transition, participants were asked to identify the location of a part on a car and metrics such as time to identify, confidence, accuracy, and cybersickness symptoms were recorded. After completing the experimental task, participants were given a short survey asking them to compare and contrast the various transition styles. The results show that different styles of transitioning users to a shared view in an immersive collaborative environment can have

drastic effects on the collaborator's spatial awareness and hence the formation of a common frame of reference on which collaborations can be based. It also shows that users can have drastically different preferences about transition styles and can be resistant to using environments that do not provide an array of transition styles. Specifically it was shown that flying users along a path to a shared view location provides significantly improved locational awareness over teleporting them directly there while requiring similar total times for reorientation. However, the cybersickness levels induced by this style can vary widely. Finally, this study provides continued motivation for additional research into VR as a collaboration environment.

CHAPTER 5. COMPARISON OF COMMUNICATION IN A MULTI-USER VIRTUAL ENVIRONMENT VS VIDEO CONFERENCING

5.1 Introduction

As discussed by many researchers, communication during the engineering design process can be one of the biggest challenges faced by a design team [259]. This problem is further compounded by the realities of a global economy which may require the design team to work with team members of suppliers who are not always present physically [1]. These distributed design teams rely heavily on a variety of modern communication tools to create a cohesive workable design, however the collaborative environment that currently exists is much more limited compared to that of the 1960s and 1970s when designers could hold an impromptu meeting around a physical design artifact, such a drawing or scale model, to share a new idea or address a problem (see Figure 5.1). Today's design challenges have also become more complex, requiring more sophisticated tools to solve, and producing more complicated 3-dimensional (3D) designs [259]. However, these distributed design teams are often limited to communicating with remote members via modern video conferencing software such as Skype or WebEx. At their best these systems typically use a shared 2D view of one participant's screen or application and a single mouse pointer for all the participants to share.

Virtual Reality (VR) has been investigated as a solution to various engineering design problems as early as 1993 [260], and even to improve engineering communication and collaboration as early as 2000 [51]. However, these early systems utilized technology which is expensive and incurs high overhead costs to run making it prohibitively expensive for companies to provide their design teams with general access [25]. Recently, new consumer grade VR hardware has been produced with much lower costs and fewer barriers to entry [241]. This new hardware could pave the way for more general access to VR technology and hence VR tools for design teams. This work presents a VR collaboration tool built with new consumer-grade VR hardware as well as a study

Figure 5.1: Photo of a design room in 1961 at AMC motors showing how designers could easily gather around the design artifacts for a collaborative discussion.

to compare the performance of the new VR tool against current video conferencing software. This paper will proceed as follows: Section 5.2 will review related research, Section 5.3 will provide an overview of system design and implementation, Section 5.4 will explain the methodology used in the comparitive study, and Section 5.5 discusses the results and implications of the study.

5.2 Motivation

The literature from the past 25 years records an array of investigations on using VR to improve various product and process design activities. The areas of virtual manufacturing, virtual assembly, and virtual disassembly have received some of the heaviest focus [50, 247, 261, 262], however, the use of VR as a communication tool has also long been a topic of interest and investigation [6]. This is likely due to VR's advanced visualization capabilities.

An important underpining of effective communication and collaboration is creating a shared understanding of the problem. Chellali et al. refer to this as a *Common Frame of Reference* (COFOR) [250]. Depending on the issues under discussion, language can sometimes be sufficient for establishing the COFOR. Sometimes, vocabulary must grow to provide finer distinctions. One example of this is that languages which developed in arctic areas have many more words for distinguishing snow and ice characteristics than do languages that developed in more temperate ar-

eas [263, 264]. This trend is observed as well in many sub-disciplines of engineering, often called *jargon*. However, as the engineering problems and solutions become more complex, language and lingo alone are not sufficient to fully establish a COFOR and collaborators turn to visual representations. Until the advent of modern computer-aided engineering (CAE) software in the 1980's [22], most visual representations were either 2D drawings or physical 3D models. Modern CAE tools allow designers to create digital 3D models of a particular design without having to build a physical model.

Unfortunately, these complex 3D designs are still most often viewed on a 2D computer screen which requires advanced spatial reasoning skills to piece together a 3D mental model of the design [105]. Past studies with VR have shown that people are able to understand these complex 3D designs faster and more completely in an immersive virtual environment (VE) where they can see the design in 3D. Satter and Bulter showed that users in an immersive stereoscopic display were more adept at navigating the environment as well as finding and repairing errors inside the environment [26]. Berg and Vance showed that a design team using an immersive VR environemnt for three design reviews were able to identify issues and propose solutions not found or solved with traditional CAE tools [247]. In addition the design team commented that the immersive environment encouraged and increased team engagement. Bochenek and Ragusa found that design reviews which included stake holders, such as the end consumer, were more successful when held in an immersive VE than in a more traditional 2D presentation [25].

In 2000, General Motors in conjunction with HRL Laboratories created a system for Distributed Design Review in Virtual Environments termed *DDRIVE* in order to leverage the benefits of VR for remote collaboration [51]. DDRIVE linked the Cave Automatic Virtual Environment (CAVE) at HRL with the CAVE at GM and the CAVE at the University of Illinois. It also allowed people to participate in the distributed design review via a more standard workstation, but only in a 2D manner. While this project proved successful, Daily et al. found that the network capacity of the time was a limiting factor [51]. Bochenek and Ragusa also point out that given the high cost of traditional immersive VR hardware, careful cost/benefits analyses need to be performed for potential application [25].

Twenty-five years later, network bandwidth has improved significantly. However, traditional CAVE setups are still large capital investments for companies and hence have not seen

widespread adoption or availability, limiting their potential impact. Recently, new consumer grade VR hardware has been introduced to the market which provides an immersive experience similar to a CAVE for ansingle person. However, unlike a traditional CAVE, this new hardware is significantly cheaper and easier to deploy requiring little to no modification to existing spaces [241].

Given the advantages immersive VR has shown previously in the literature, this new low-cost consumer VR hardware could prove an important enabler for companies and design teams to begin leveraging VR CAE tools in more consistent daily tasks. However, in contrast to a traditional CAVE where a group can use the CAVE together and see each other, the currently available consumer VR hardware is in the form of a head-mounted display (HMD) which only a single participant can use at a time and their view of the outside world is blocked. This work demonstrates a system where multiple users can enter a shared immersive VE, with each participant in their own headset connected via a network. The VE also includes gesture support to create a collaborative environment similar to that of a CAVE. This work also presents a study that was performed on the system to evaluate the system's effectiveness as a communication tool compared to commonly used tools such as Skype.

5.3 System Architecture

5.3.1 Overview

At a high level, the system is built on a client-server architecture as shown in Figure 5.2. Each client is made up of a single VR hardware set (discussed in detail in Section 5.3.2) and a client program which exchanges various data packets with the server. In this way, the server can distribute the local data from each client to the other clients in the session. A client can then use the information it receives from the server to update its local version of the VE and animate avatars for each participant. This architecture provides a way for creating a syncronized environment for each participant regardless of where the participant is located. Hence, all participants could be physically present (as would be required for a CAVE) or spread across the country.

While this architecture could support sending data from a variety of inputs (such as motion capture systems, controllers, HMDs, position sensors, etc.) the hardware included in the implementation evaluated here was limited to VR HMDs and motion capture systems for the hands and

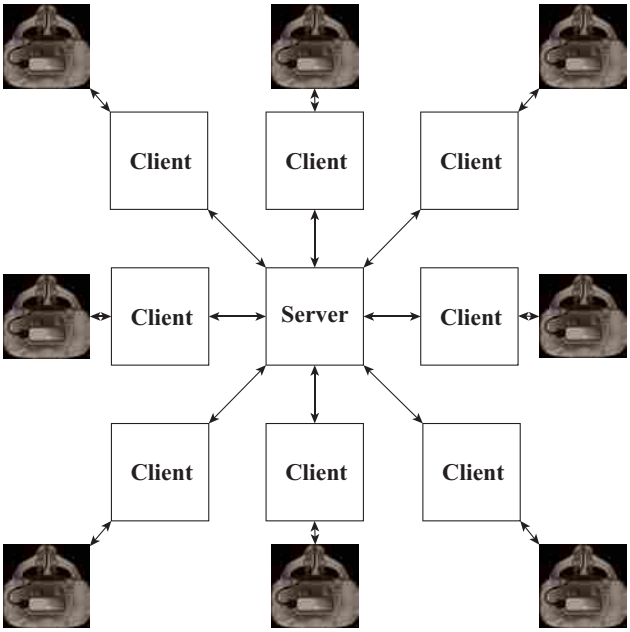


Figure 5.2: System Architecture showing how multiple single-user Head-Mounted Displays can be linked through a client-server architecture to create a shared immersive environment.

fingers. This hardware allow the system to create a set of virtual hands for each user as their avatar and send the appropriate data across the network the recreate each user's hands in the VE. This allowed participants in the VE to see the hands of every other participant and see where or what another participant might be pointing or gesturing to. An example of this is shown in Figure 5.3 With a more complete motion capture system, more data about the user could be captured and used to create and animate higher fidelity avatars including items such as legs, arms, and a torso. Technology such as that demonstrated by Li et al. could be used to track facial expressions and animate the face of the avatar to match the human participant it represents [95].

While streaming audio through a network is also possible and has been solved many times, playing audio in virtual reality so that it feels like a cohesive part of the VE is a non-trivial issue which has been the focus of much research. The main issue is the unique shape of each person's ears which alter the waveforms of the external sounds we hear and gives our brain additional clues to help deduce the location of the sound's origin [144]. Currently measuring the unique audio signature of each person's ear is a time consuming task requiring specialized hardware. However, there are currently plans to commercialize a much quicker process which would make the technology more accessible [146]. Given the issues surrounding virtual audio and the fact

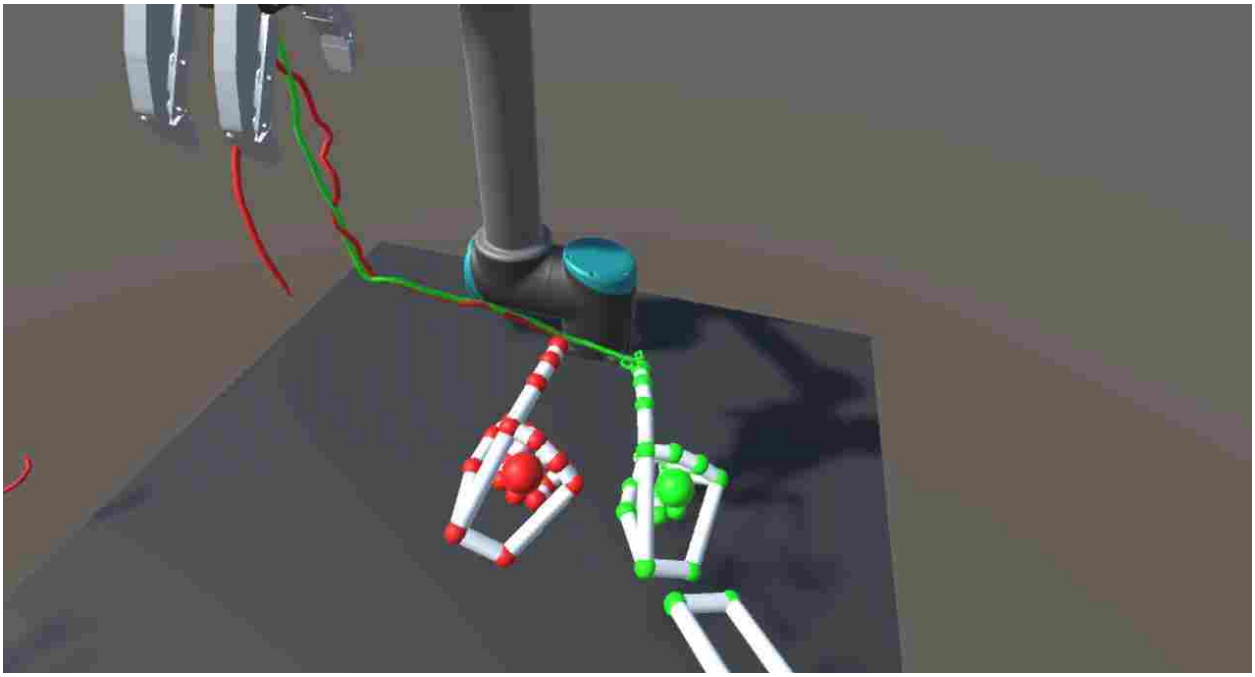


Figure 5.3: An example of a local user (green hand on the right) teaching a remote user (red hand on the left). Both users can see each other's hands and gestures improving the communication.

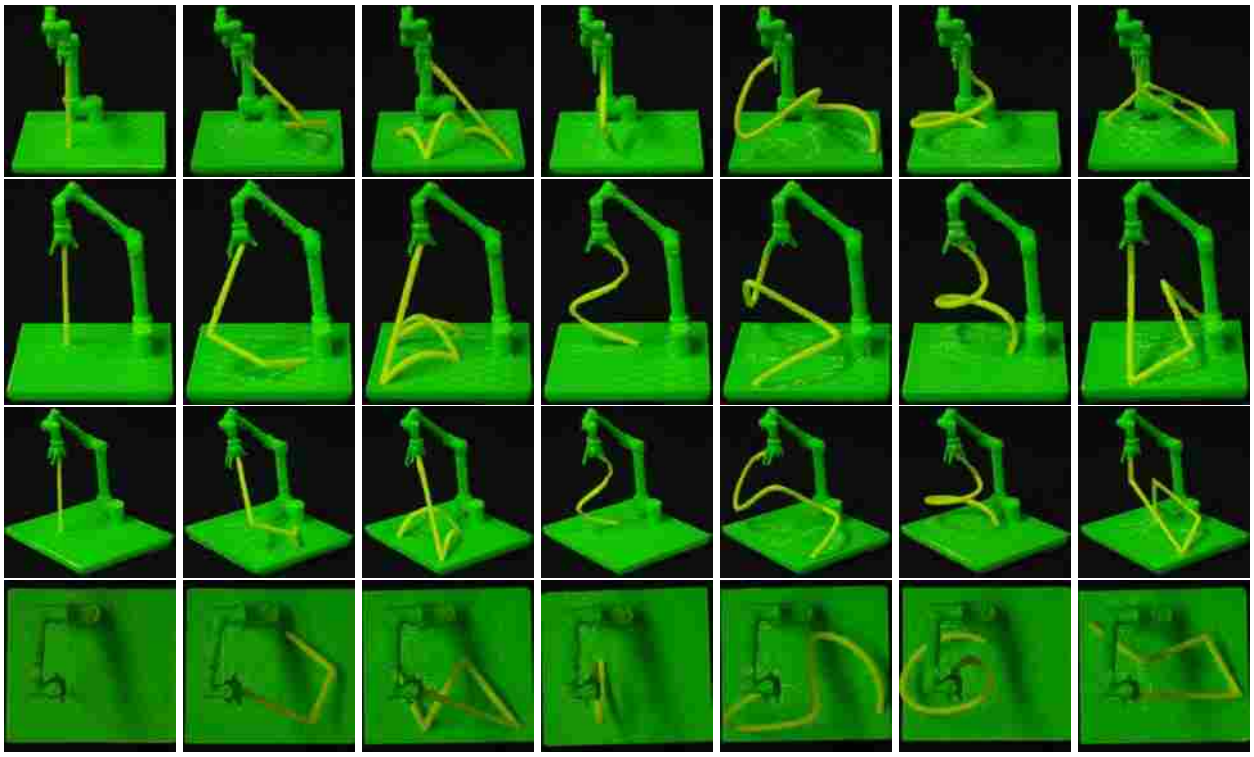
that our implementation was set up such that users would be in close physical proximity, audio streaming was considered outside the scope and not implemented.

In addition to creating an avatar for each participant, design artifacts such as 3D models were added to the VE. In combination with the avatars, this allows a group of collaborators to meet around a 3D design artifact and see each collaborator's gestures (such as pointing to a location, indicating a size, or showing the movement of a mechanism) in relation to the model. Additionally, the system recognised special input gestures which allowed a collaborator to draw and delete 3D free-form curves. This functionality provides collaborators a 3D sketch-pad to quickly communicate new ideas, modifications to a design, or even to highlight and annotate design artifacts in the VE.

5.3.2 Hardware

At a minimum each client needs a VR HMD and a computer with a network connection to power the HMD and connect to the sever. In this implementation HTC Vive headsets were used in conjunction with upgraded Windows workstation computers (3.5GHz, 8GB RAM, NVidia Geforce

Figure 5.4: HTC Vive HMD with front mounted Leap Motion™ Controller. Courtesy of the Leap Motion™ Blog.



Path 1 Path 2 Path 3 Path 4 Path 5 Path 6 Tutorial

Figure 5.5: Front, Side, Isometric, and Top views of the paths used in the study.

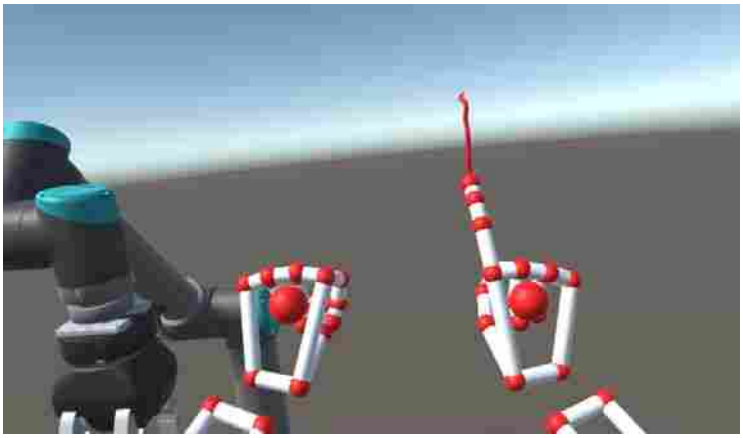
1060 GTX). Additional sensors and inputs can be added to increase the fidelity and functionality of the VE. For the system presented, motion capture of the hands and fingers was deemed sufficient and a Leap Motion™ Controller was used to provide this functionality. The Leap Motion™ Controller was mounted to the Vive HMD and connected to the computer via USB 3.0 as shown in Figure 5.4.

The computer used for the server was a standard workstation computer (2.66GHz, 12GB RAM). Both the server and the client machines were connected through the local college network with 10 Mbps or greater network connection speeds.

5.3.3 Software

Software development was done in Unity® (version 5.4) using standard assets and libraries where possible. Most notably, Unity's built-in networking libraries and assets were leveraged to provide the backbone of the networking communication. Where standard libraries and assets were not sufficient, 3rd party libraries and assets were used. In particular, libraries, assets and examples provided by Leap Motion™ were used to interface with the Leap Motion™ Controller. The Orion V3 beta drivers were used for the Leap Motion™ Controller which provide enhanced tracking, reduced drop-out, and a larger tracking area than previous drivers.

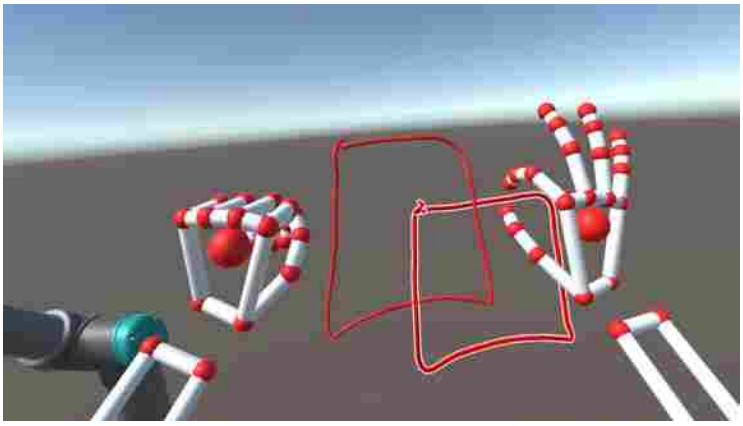
Where standard and 3rd party libraries were insufficient, additional scripts were added or existing ones modified. For example, Leap Motion™ libraries were modified to expose the raw location and orientation of the constituent pieces of the local avatar. This data was then serialized, sent over the network, and used to create an identically located and oriented avatar on the remote clients. Additionally, assets were created to recognise the special gestures that would trigger input to the system. For example, the pointing gesture, shown in Figure 5.6, (index finger extended, all other digits closed) would start drawing a 3D free-form curve and continue drawing the curve until the gesture was broken. A pinching gesture, shown in Figure 5.7, (thumb and index finger touching, middle, ring and pinky fingers extended) was used to delete a specific free-form curve across all connected clients. A gesture consisting of a double "thumbs-up", shown in Figure 5.8, (all eight fingers closed, both thumbs extended and pointed towards the ceiling) was used as a general clear which would delete all the free-form curves across all connected clients. The appropriate data for each of these input gestures was also serialized and sent across the network to replicate the effect



VR Gesture

Physical Gesture

Figure 5.6: The pointing input gesture to draw free-form curves.

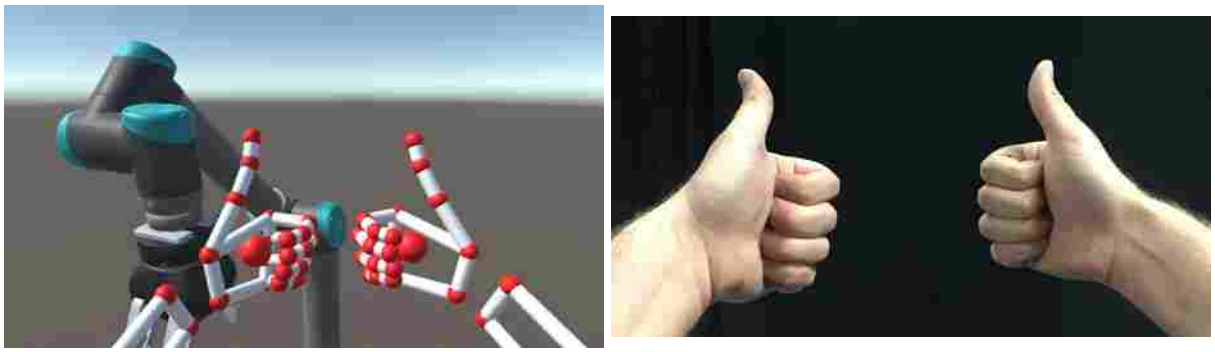


VR Gesture

Physical Gesture

Figure 5.7: The pinching input gesture to delete a free-form curve.

of the gesture on the remote clients. Hence, as a collaborator drew a free form curve locally, the curve points were serialized and sent to the other clients through the server. Once the data reached the client it was reconstructed into a free-form curve matching the original. Each curve was also uniquely named such that it could be uniquely identified on each client. Thus, when a collaborator deleted a particular curve, messages were sent across the network deleting the curve from all clients keeping the VE consistent.



VR Gesture

Physical Gesture

Figure 5.8: The thumb-up gesture to delete all free-form curves.

5.4 Methodology

5.4.1 Overview

In order to test the hypothesis that the system described in Section 5.3 would improve communication against current video conferencing tools, an experimental study was conducted to assess the change in both communication time and accuracy when using the collaborative VE or Skype to communicate 3D geometry data. It was hypothesized that a reduced amount of time to communicate would be required and that there would be increased accuracy of the information being communicated when the collaborators used the VE.

In order to test this hypothesis, participants were recruited and required to bring a friend or colleague with them to form a team of two. During the study each team participated in a single session lasting approximately 1 hour during which the two team members would be randomly assigned the roles of teacher and student. The teacher would be given a 3D printed model of the movement path of the tool head of a robotic arm (shown in Figure 5.5). After learning the path, the teacher would use either Skype or the collaborative VE described in Section 5.3 to teach their teammate the path. Once the teammate felt like they knew the path, the communication link between the team would be broken, and both team members would use the VE in a non-collaborative mode to record their understanding of the original path. This recorded data was used to analyze communication accuracy and was always recorded in the VE regardless of whether communication was done in Skype or the VE. Additionally, the time that a teacher spent teaching the path to their

Figure 5.9: The experimental setup used for the study. When using Skype, participants would sit at the computers. When using the VE, participants would stand and walk around in the white areas. A curtain between the two computers was used to block the line of sight between participants.

teammate was recorded and used to analyze the amount of time required to communicate a path in Skype vs in the VE.

5.4.2 Setup

Since this implementation of the collaborative VE does not include support for audio, the audio streaming support in Skype was not used. Instead, the physical areas for each client were placed in close proximity such that participants could speak to each other normally as seen in Figure 5.9. However, when communicating the line of sight between participants was blocked such that all visual communication went through the appropriate collaboration tool.

Six unique paths were created with varying degrees of complexity. The 3D printed models that were given to the teacher to memorize are shown in Figure 5.5. Path 1 was one dimensional consisting of a straight line. Path 2 was 3D consisting of three straight lines. Path 3 was 3D consisting of one straight line and three similar arcs. Path 4 was 2D consisting of one "S" shaped curve. Path 5 was 3D consisting of 1 long free-form curve. Path 6 was 3D consisting of one

stylized spiral. The tutorial path was also 3D consisting of five straight line segments. The tutorial path was used for explanation and training purposes.

Of those six paths each team was assigned to teach four paths; two in the VE and two in Skype. The order and collaboration environment were randomly assigned such that each model was taught by 24 teams, 12 times in the VE and 12 times in Skype. In addition the order in which the environments were used was also randomized such that half of the teams used the VE first and half of the teams used Skype first. The roles of teacher and student were also randomized such that each participant taught once in Skype and once in the VE.

5.4.3 Experimental Session

Each team was asked to participate in a single one hour session. When a team arrived they were given an overview of the experiment, and signed the appropriate waivers. After signing the waivers, the participants were given an overview of the Sykpe collaboration environment and the tools available. After explaining Skype, the team was given an overview of the VE, how the headsets and motion capture worked, and the tools available for collaboration in the VE. After the team had been familiarized with both systems, they were given time to practice with the VE and the input gestures to draw and delete free-form curves.

After the familiarization phase, participants were given a tutorial on how their final input would be recorded for accuracy analysis. During the tutorial, the tutorial path was placed in the VE and participants were asked to trace it. Each participants' trace of the tutorial path was recorded and used as baseline accuracy data. An example is shown in Figure 5.10

After participants had completed familiarization and the tutorial, the experimental task begun. One teammate was randomly assigned to be the teacher and given a model to memorize. After memorization, both teammates were set-up in the randomly assigned collaboration environment. A timer was started when the teacher began the explanation, and was stopped once the student felt they could reproduce the path. After the timer was stopped, no further communication or clarification was allowed. In order to keep experimental sessions under one hour, the teaching time was limited to 4 minutes. After the explanation, participants used the VE in a non-collaborative mode to record their understanding of the path. The full experimental task (starting with random teacher selection) was repeated until the team had taught all four of their assigned paths.

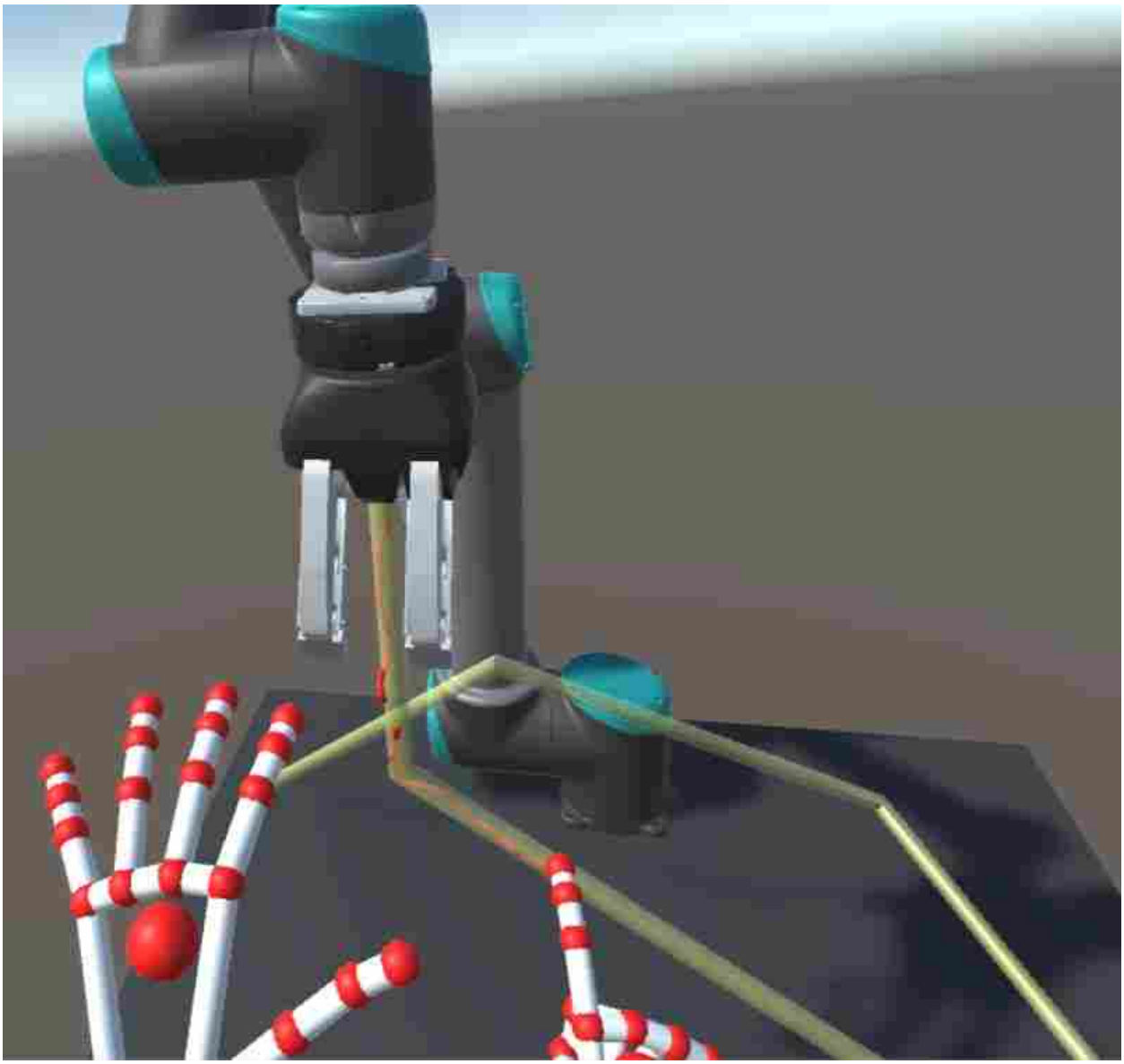


Figure 5.10: An example of a participant tracing the tutorial path in the VE.

After teaching all four paths both team members were asked to complete a short survey about their experience with the tool. This survey completed an experimental session.

In order to more closely approximate a real-life collaboration situation, participants had access to a 3D model of the robotic arm and tool head in both Skype and the VE. In Skype they had access to the model through the Autodesk A360 viewer. This application provides standard viewing controls such as pan, rotate, and zoom. Using Skype, participants were able to share their screen and show their teammate their view of the model and indicate locations they were

referencing with their mouse. For the Skype environment, webcams were also provided so that participants could also video chat and use their hands and arms to demonstrate the path if preferred. A model of the robotic arm and tool head were placed in the VE as well, scaled to full size. In the VE participants were able to use their hands (as virtual avatars) to point, touch, and indicate locations on the model. In addition participants could use the drawing tools to also mark, highlight, and draw in the VE.

5.4.4 Participants

74 participants were recruited from Brigham Young University's School of Engineering and Technology to form 37 teams. 36 datasets were collected with the 37th team used to replace a corrupted dataset from an earlier team. Participants ranged from college freshmen to graduate students. Of the 74 participants, 58 had less than 5 hours previous experience with VR and only 6 had more than 10 hours of previous experience. In order to encourage participation, each participant was compensated $10 for completing the one-hour session. In order to encourage participants to explain the path clearly and quickly, participants were measured on the speed of their teaching and the accuracy of their response paths. To incentivize quick, high quality work, the two top performing teams were awarded VR headsets and gift cards. Although participants were allowed to terminate their participation early for any reason, none did.

5.5 Results & Discussion

5.5.1 Data Processing

In order to measure how well a participant understands a particular path, each participant was asked to record the path in the VE. From each recording, the defining points of each curve were logged to a file along with information about which curves have been deleted. In order to analyze this data, the raw point data for each curve that was not deleted by the user was imported into Siemens® NX 9. Then, extraneous curves, redundant curves, overlapping portions, and tag ends (unintentional curve starts and stops) were removed. The Fit Spline tool in NX 9 was then used to create a smooth spline through the cleaned points for each curve. After a spline was fit to

Table 5.1: Mean time to teach a path by Environment and Model, percent reduction of mean time in VE over Skype, and p-values from a Student's t-test of means. Bold values indicate significance at the 90% confidence level.

Model	1	2	3	4	5	6
Skype Mean Time (s)	17.17	167.95	168.61	121.06	172.95	129.69
VE Mean Time (s)	26.64	98.71	92.19	82.00	108.47	102.20
Percent Reduction	-55.15	41.22	45.32	32.27	37.28	21.20
p-value	.911	**.008**	**.001**	**.0691**	**.002**	.1832

the cleaned points, 100 equidistant measurement points were created along the spline (as well as the original model paths). To calculate how accurately one curve matched another, the distance (in mm) between corresponding points (i.e. point 1 to point 1, point 2 to point 2) was calculated and summed over all 100 measurement points. For each team, the accuracy of the teacher's curve to the original model, the accuracy of the student's curve to the original model, and the accuracy of the student's curve to the teacher's curve were calculated.

We note here that the accuracy calculations used in this work are concerned mainly with the accuracy and precision of the path and not conceptual understanding. A good conceptual understanding of the path is important for being able to accurately and precisely record the path, however, a good conceptual understanding of the path does not guarantee an accurate recording of it. Gauging conceptual understanding of a path is a much more difficult problem, however, there has been work by Wobbrock et al. [265] and others [266, 267] that might be extendable to better quantify the conceptual understanding of a path.

5.5.2 Results

As shown in Table 5.1 a Student's t-test shows statistically significant reductions in the amount of time required teach a path for Paths 2–5. The time reduction for Path 4 is interesting since Path 4 is two dimensional and hence an environment such as Skype should be sufficient to fully convey the information. Also of interest is the fact that, despite being a 3D path, the time to teach Path 6 was not statistically different. This result may in part be explained by the fact that Path 6 is a spiral and hence it was easy to describe the general shape of the path verbally.

Table 5.2: Mean Accuracy by Environment and path. P-values from a Student's t-test. Bold values indicate significance at the 90% confidence level.

Comparison	Skype Acc.	VE Acc.	p-value
Path 1			
Teacher–Model	3476	3221	.318
Student–Model	3832	3071	**.094**
Student–Teacher	4585	3923	**.079**
Path 2			
Teacher – Model	14112	10867	.237
Student – Model	15926	12126	.394
Student – Teacher	17883	11844	.111
Path 3			
Teacher – Model	11581	10344	.200
Student – Model	16488	12516	**.034**
Student – Teacher	15961	11858	**.042**
Path 4			
Teacher – Model	10673	8107	**.065**
Student – Model	11777	9210	**.081**
Student – Teacher	12586	10247	.152
Path 5			
Teacher – Model	16215	19041	.882
Student – Model	21512	20919	.426
Student – Teacher	17309	11689	**.002**
Path 6			
Teacher – Model	14040	14618	.596
Student – Model	18082	17030	.368
Student – Teacher	16788	14104	.168

Finally, although not statistically significant, it is also interesting to note that, on average, Path 1 took longer to explain in the VE than in Skype. This result likely stems from the relative simplicity of teaching Path 1 and the fact that in the VE participants had trouble with the sensor not picking up an input gesture. Hence, they would spend several seconds adjusting their pose until the sensor recognized the gesture. Although these delays existed in all the VE time measurements, only Path 1 showed a negative effect due to the extremely short time required to teach it.

In Table 5.2 the mean accuracy values are calculated as described in Section 5.5.1 as well as p-values from a Student's t-test means comparison. From Table 5.2 it can be seen that there was a statistically significant improvement in accuracy of the student's recording to the teacher's recording of Paths 1, 3, and 5. Paths 1, 3, and 4 also had statistically significant improvements in the accuracy of the student's recording to the original model (which the students never saw). It is interesting to note that Path 4 shows a statistically significant improvement in the teacher's accuracy to the original model, and in general (though not statistically significant with the current amount of data) this trend holds for most of the paths. This could potentially be explained by the teachers who taught in the VE having additional practice drawing the path in the VE, the fact that teachers who taught in the VE did not have to fit their mental model to 2D views of the 3D model, the fact that teachers in the VE had a shorter time between memorizing the model and recording their answer, or by some combination of these factors. Path 5 is also interesting due to the student's accuracy to the original not being significantly different, while their accuracy to the teacher is better. This is likely caused by the difficulty of memorizing Path 5. During the study, it was observed that parts of Path 5 were frequently forgotten or taught incorrectly regardless of the system being used for collaboration.

Figure 5.11 shows the frequency of participants' responses to the question of how suitable each environment was for communicating complex 3D information. As shown in Figure 5.11, Skype was typically rated as Somewhat Suitable while the VE was typically rated as Ideal or Very Suitable. Table 5.3 shows the frequency of various positive and negative reasons given when asked why they assigned each environment a particular suitability rating. The most common reason for rating Skype's suitability lower than the VE's was that using Skype only provided a 2D view of the 3D information requiring the student to extrapolate 3D position and shape from multiple 2D views where as the VE allows native 3D viewing. Additional common reasons for rating the VE as more suitable included the ability to draw and annotate the 3D model in the VE and the ability to gesture (point, sweep, demarcate, etc.) with their hands in 3D instead of using a 2D mouse pointer in Skype. Reasons for marking Skype higher on the suitability scale included the ability to see the other person's face in Skype and the high precision of the mouse input. The most common reason for marking the VE lower on the suitability scale was the difficulty some participants had getting the VE to recognise their input gestures to draw or delete curves.

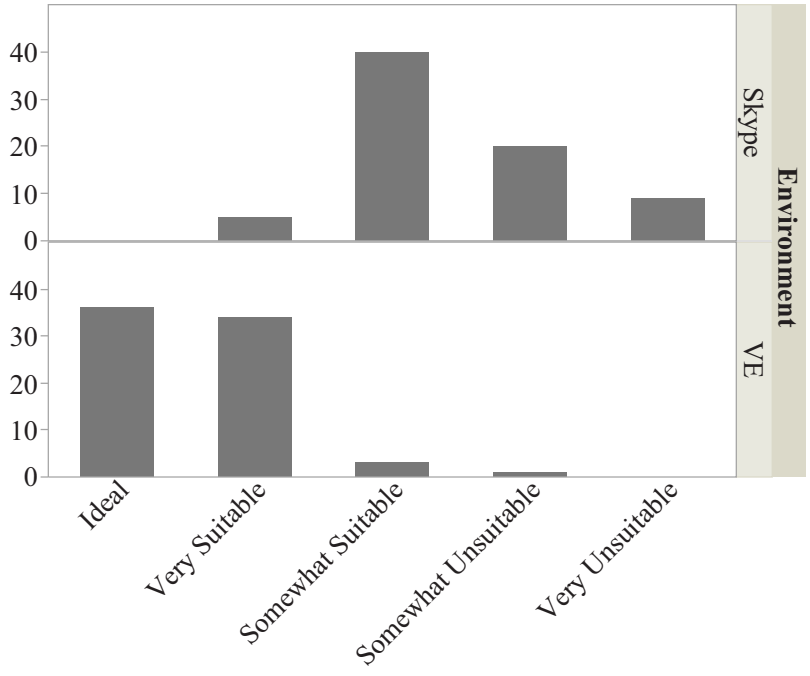


Figure 5.11: Frequency of responses of how suitable each environment was to communicating complex 3D data.

Table 5.3: Frequency of Reasons Given for the Suitability of Each Environment for Communicating 3D Information

Skype	*Pros*	Reason
	7	Could see the other person
	6	Precision from Mouse Input
	Cons	
	32	Viewing was 2D
	13	Communication less detailed
	3	Difficulty controlling software
VE	*Pros*	
	36	Could View in 3D
	26	Could Draw
	25	Could Gesture (point, etc.)
	16	More Interactive
	5	Simultaneous Interaction
	Cons	
	7	Poor Input Gesture Recognition

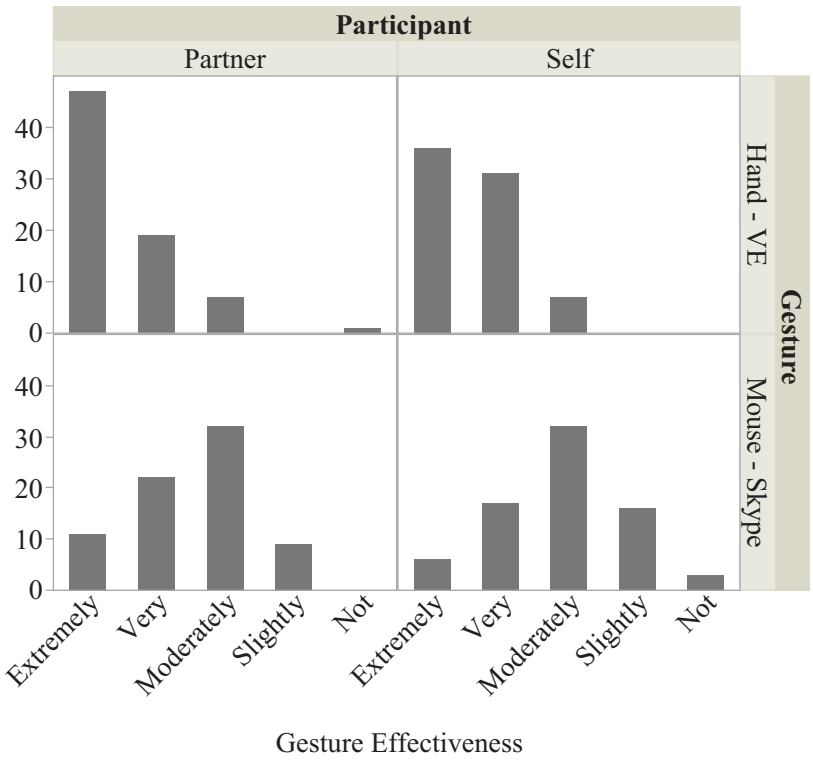


Figure 5.12: Frequency response of how effective various gestures types were at conveying complex 3D information. Broken out by a participant's self rating and rating of partner.

Figure 5.12 shows the frequency of responses of how effective participants felt either mouse gestures in Skype or hand gestures in the VE were for conveying the path information. Responses are also broken out by the effectiveness of the participant's own gestures vs the effectiveness of the partner's gestures. It is interesting to note that participants typically rated their own gestures as less effective than their partner's regardless of the system. However, despite this self-deprecation, participants felt their own hand gestures were significantly more effective at communicating the path to their partner then their partner's mouse gestures in Skype.

Figure 5.13 shows participants' environment preferences for communicating complex 3D data. From their responses we see that 69 out of the 74 respondents (93%) prefer the VE, 1 (1%) prefers Skype and 4 (5%) said they would prefer something else such as showing in person on a physical model or drawing with pen and paper.

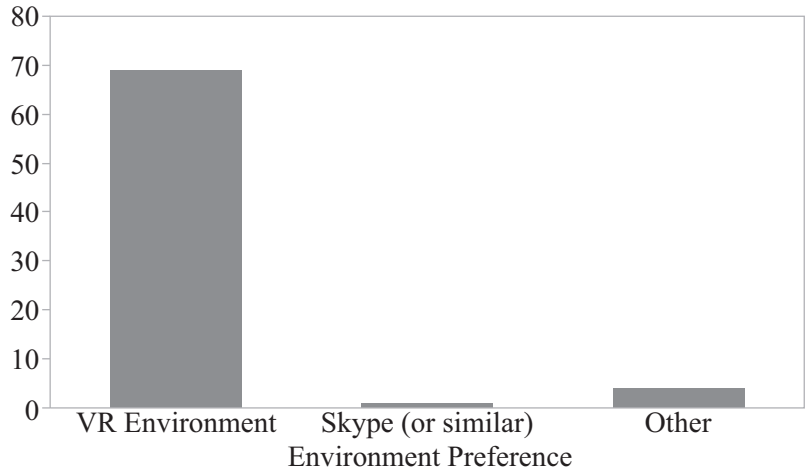


Figure 5.13: Frequency response of participants' environment preference to communicate complex 3D data.

5.5.3 Discussion

Taking together the results presented above, in many cases, a collaborative VE that allows for gestures and communication tools can provide significant collaboration benefits over current video conferencing systems, and at much more reasonable costs than similarly capable CAVE systems. In many situations, even when the information is only 2D, there are improvements to both the amount of time required to communicate and the accuracy of communication. From the participant's feedback this appears to be due to 3D viewing instead of 2D, communication tools such as drawing, and support for communication gestures such as pointing and waving. In addition to the results presented above, 96% of participants stated that they enjoyed using the VE, 90% believed that VR tools could improve their engineering work, and 86% of participants stated they would like to have access to VR tools in their future workplace. While these survey responses may not be representative of the larger population due to the volunteer nature of the participants, the results are sufficiently consistent to not be ignored completely.

Limitations

There are two significant limitations to this work. First and foremost as mentioned previously, the participants in this study were recruited on a volunteer basis. This means that the sample is not well representative of the general population, and may have a higher predisposition to prefer

VR. However, given the more wide-spread interest and commercial competition in VR and augmented reality products, this predisposition likely does not differ significantly from the general population of a similar age. In addition, it is this younger population that forms the future of the engineering workforce.

Secondly, this work did not include audio streaming. Before a collaborative VE could fully support a physically distributed workforce, audio streaming would need to be implemented and tested. Depending on the specifics of the network connecting the system, a noticeable delay could be introduced which could hamper collaboration effectiveness. However, this delay would be present regardless of whether the collaborators were using a video conference or a collaborative VE. In addition, since more of the information needed to be conveyed verbally in the video-call environment, network audio delays may be more detrimental to the video-call environment than to a collaborative VE.

5.6 Conclusions and Future Work

As discussed in Section 5.2 the literature is replete with potential applications and proven benefits to VR systems especially as collaborative tools. However, until the recent release of low-cost consumer-grade VR hardware, VR setups were prohibitively expensive and hence use of the hardware was limited to the most important tasks. Now, VR hardware has reached a tipping point where deploying VR hardware to large portions of a company's workforce is more feasible. In turn, this higher availability would make a collaborative VE, such as the one presented here, not just feasible, but beneficial by improving communication, collaboration, and design understanding. These improvements could reduce the number of costly turn backs and design changes required after a design freeze. Long term, distributed collaborative VEs could lay the foundation for a globally distributed workforce which is strengthened by talents, values, and expertise of many cultures and countries but allows them all to work together as if they were located locally. In fact, widespread availability of VR and the related technologies of augmented reality could produce as monumental of changes to our global society as the invention of the airplane, the computer, the Internet, and smart devices; all of which have significantly changed and improved how humans communicate, collaborate, design, engineer, and manufacture.

5.6.1 Future Work

While this work has demonstrated the feasibility and benefits of a collaborative VE using low-cost consumer-grade hardware, additional needed investigations remain. When asked what improvements and changes participants would like to see in the VE, the most common request was improved hand tracking for gestural input such as drawing or deleting curves. Additional common requests included an expanded toolbox such as color selection, points, straight lines, grid snapping, and shape primitives, the ability to modify and delete portions of previously created curves, the use of a more precise input such as the Vive controllers for drawing curves, the ability to more actively control your viewpoint such as zooming in and out, and improved graphics.

Additional future work also includes the integration of additional sensors to create a richer more immersive experience. Additional sensors to integrate include full body motion capture which could be used to create full body avatars, real-time facial capture which would allow animation of the avatar's face to convey an additional depth of communication, and as mentioned previously, integration of 3D audio streaming to allow the system to be distributed.

CHAPTER 6. CONCLUSION

As discussed in Chapter 2 the primary object of this research has been to answer questions fundamental to the creation of a Collabortive Virtual Environment (CVE) for Distributed Design Reviews (DDR) using low-cost consumer-grade Virtual Reality (VR) hardware. An overview of the findings from this research is available in Table 6.1. In order to answer these questions the research was split into three portions.

The first portion of this dissertation research which is presented in Chapter 3 consists of a review of the recently available low-cost consumer-grade hardware and included devices from stereo Head Mounted Displays (HMDs), Motion Capture, input controllers, audio processing, olfactory displays, and gustatory displays. It was found that much of this hardware has reached a quality and cost tipping point where companies can begin to invest in providing more general access to VR hardware for their workforce. The quality of these devices is sufficient to support a high percentage of the applications in the existing literature previously developed with much more expensive VR hardware. In addition, a review of these previously published applications was performed. It was found that the Preliminary Design, Detailed Design, and Producibility Refinement stages of the Design Process have seen the heaviest focus for improvement with VR applications. However, it was also found that all stages of the Engineering Design Process, and even the entire Product Life-cycle have the potential to benefit from VR applications. From these findings it is concluded that the new low-cost consumer-grade hardware currently available on the market could significantly affect how engineering design work is performed.

The second portion of this dissertation research which is presented in Chapter 4 consists of an experimental research study to evaluate the trade-off between the amount of cybersickness and disorientation induced by various styles of creating a Shared Context in a CVE. In addition, user preference was surveyed and analyzed. Since the virtual space in a CVE could be much larger than the physical space a user has to work in, it is important to find a way to move users to a

Table 6.1: Summary of Research Results

Research Question	Hypothesis	Result
Hardware Review		
What consumer grade hardware is currently available?	N/A	New HMDs & Motion Capture
What has changed in VR hardware since 2000?	N/A	Affordability & Capability
Which major areas of the Engineering Design Process have been the focus of previous VR research?	N/A	All
How could this new low-cost consumer-grade hardware affect the design process going forward?	N/A	Pervasive Daily Use
Shared Context Creation		
What is the trade-off between cybersickness and disorientation for each style of Shared Context creation?	N/A	Various
How much disorientation results from a teleportation style context creation?	High Amount	Moderate
How much cybersickness results from a teleportation style context creation?	Low	Low
What preference do users have about creation style?	Fly	Fly
What preference do users have about the trade-off between cybersickness and disorientation?	Reduce Disorientation	Reduced Disorientation
Which style of creation allows the quickest identification of the Shared Context location?	Fly	Teleport, Fly
Which style of creation has the fewest misidentifications of the Shared Context Location?	Fly	Manual
How severe is the cybersickness experienced during Shared Context creation?	Minor	Mild to Moderate
CVE with Gestural Support		
Does a CVE with gesture support allow quicker communication of 3D information?	Yes	Yes
Does a CVE with gesture support allow more accurate communication of 3D information?	Yes	Yes
Do a majority of users prefer the CVE over legacy systems?	Yes	Yes
Do a majority of users find gesturing in a CVE a natural way to communicate 3D information?	Yes	Yes
Do a majority of users find the virtual replication of gestures recognizable and understandable?	Yes	Yes

Shared Context Location in a way that allows them to collaborate effectively. It was found that teleportation (which is currently the predominant movement style in single-user VR applications) induced low amounts of cybersickness in participants. However, participants generally reported

that teleportation caused the highest amount of disorientation of any of the creation styles studied. The majority of participants preferred the Fly creation style where by they were automatically moved along a smooth path from their current location in the VE to the Shared Context location allowing them to see where they had been moved in the VE. Participants had mixed opinions on the amount of cybersickness caused by Fly compared to that caused by Teleportation. The responses formed a bimodal distribution with the major mode reporting higher amounts of cybersickness and the minor mode reporting lower amounts of cybersickness. Regardless, most participants reported significantly lower levels of disorientation with Fly, and given that it was the preferred method it is concluded that a majority of participants prefer a trade-off which potentially increases cybersickness slightly while significantly reducing disorientation. Interestingly, it was also found that approximately one quarter of participants said they would not use a CVE where the sole method for creating a Shared Context was Fly. This finding highlights the need to provide the user multiple creation styles so they can choose how to make the trade-off themselves.

The third portion of this dissertation research, presented in Chapter 5, consists of the development of a CVE with gesture support and a comparative study of communication in the CVE vs communication in a video conference (which is the predominately used remote collaboration environment). This research demonstrates that it is possible to create a CVE with gesture support using currently available low-cost consumer-grade VR hardware by connecting each set of hardware through a network via a client-server architecture. The comparative study showed that in certain cases participants were able to communicate up to 45% faster than via a video conference. In addition to being able to communicate faster, participants were able to convey some 3D paths more accurately than through the video conference. Additionally, the study found that participants were able to understand gestures made in the CVE significantly better than motions made with the mouse in the video conference, and more than 90% of participants preferred the CVE over the video conference for communicating complex 3D information. These results show that a CVE with gesture support can reasonably be built from currently available low-cost hardware and that it has measurable communication advantages over current collaboration environments.

Summing the results from this research, it can be concluded that VR hardware has reached a tipping point where it has become much more accessible and able to support many of the applications developed on more expensive hardware. This unprecedented availability will allow

companies and engineers to begin accessing the benefits described from these applications more pervasively in the design process. It has also been shown that these devices can be linked together to form a cohesive CVE experience which can provide engineers the means to leverage the power of modern and next generation design tools in a fully collaborative environment reminiscent of the collaborative design environments of the 1960s. In much the same way that technologies such as telephones, airplanes, computers, the Internet, high speed networks, and mobile devices have improved the design process and made possible design methodologies never before imagined, highly accessible VR technology has the potential to enable engineering design tools, environments, and methodologies which have not yet been conceived.

6.1 Industry Implementation

The results presented from this research show compelling evidence for applying VR technology to the design process, however, it is important to understand the assumptions underlying this research. Because of the significantly improved cost-to-benefit ratio of this new hardware discussed in Section 3.5 it was assumed that in the future, engineers and designers will have wide spread access to this hardware to the point where they may use it on a daily basis for long periods of time. It was also assumed that users may spend significant time on individual design activities with-in the CVE, and would need a way to quickly gather to a specified location for a collaborative activity, hence the research presented in Chapter 4. However, in such an environment it would be disorienting, distracting, and counter-productive to be instantly pulled to a collaborative design activity at the whim of another user. Therefore, in practice, we suggest allowing the user to indicate their availability for collaborative activities and be allowed to confirm their desire to join a collaborative activity before being moved to the shared location.

Also, based on the assumptions listed above, the CVE presented in the work was designed with the intention of providing an environment which allows users to easily flow between collaborative and individual design tasks as might be expected during a typical workday. However, there are times when this flexibility is less desired such as when users are in a formal meeting, or working to finish a particular task before a deadline. In these cases it would be advisable to implement additional tools and restrictions on each client regarding what information is shown in their virtual environment.

6.2 Limitations

While this research presents compelling evidence for the widespread adoption of VR in the design process and the potential impact it would have, there are limitations of which the reader should be made aware. The most important of these concerns applicability of the study results to the general public. The sample population for both of these studies consisted of primarily of college students pursuing a technical degree, which is not representative of the general population. However, this sample of the population was selected as the primary focus since they have not yet developed strong opinions about particular work paradigms or software tools. In addition, this population represents the future engineers and designers and the scope of this research was interested in studying their preferences and reactions to VR technology.

Related to the aforementioned focus, VR technology is still considered a novelty experience, as evidenced by the study participants' amount of previous experience with the technology, and many participants were excited about the experience regardless of any real benefit from the environment. Given this general attitude toward VR technology with-in sample population, some of the survey responses may be biased in favor of the collaborative virtual environment. However, despite this potential bias in the survey responses, the non-survey results also showed statistically significant improvements and support the conclusions drawn.

Another important limitation of this work is that for some of the statistical analyses a 90% confidence level was used instead of a more typical 95% confidence level. This decision was made due to the fact that the data came from human participants which typically have much higher amounts of variability than controlled scientific processes. In the case of our statistical analysis, it is expected that larger sample sizes would show similar results with higher confidence.

6.3 Future Work

One of the important points from Chapter 3 is that this VR hardware is still in its relative infancy and much work still remains to be done to fully apply the capabilities of this hardware to the engineering design process. Here we discuss recommended follow-on work to this research.

6.3.1 Collaborative Virtual Environment Enhancement

As was noted in Chapter 3 new hardware with enhanced capabilities is beginning to emerge. The most noted positive feature of the video conferencing environment from the survey in Chapter 5 was the ability to see the other person's face and hence their facial expressions. Research highlighted in Section 3.3.1 discusses technology and hardware to support real-time facial tracking and the animation of the face of a virtual avatar based on the user's face. Providing support for this enhancement to the CVE presented in this work would allow similar viewing of each user's face from with-in the CVE potentially improving communication further.

Additionally, a commonly requested improvement to the CVE was improved drawing controls. Because of inaccuracies in the sensor as well as the form of the human body, participants in the study had great difficulty drawing straight lines and suggested that improved features for drawing in general might further improve the usefulness of the CVE. Support for drawing a straight line by specifying endpoints was common, however support for primitive 3D shapes such as cubes, spheres, ellipsoids, cones, etc, and control of appearance, such as color, pattern, and thickness were also requested. Implementation of these advanced drawing tools could allow for communication of early design ideas when no virtual or physical prototypes exist.

Research by Ian Freeman et al. has also provided ways to interact with a CAD model directly from a virtual environment while preserving much of the meta-data [268]. Adding support for these features to the CVE could also improve collaboration by increasing the control users have over the design artifacts in the CVE. If such a methodology could be extended to include the joints and constraints on the virtual models, users would then interact with these mechanisms in the CVE to better understand and explain their motion.

In the survey from the study presented in Chapter 4, participants often asked for an improved set of manual controls. While there is a wealth of research studying active navigation techniques in immersive environments, the Navidget [269], "World in Miniture" with previews [254], and speed coupled flying with orbiting [270] seem particularly promising. Some participants mentioned that a better set of manual controls might change their preference. In addition, providing information about where the Shared Context was being established may help users to navigate to the location more quickly, mitigating the speed advantage of the automatic styles. This merits further investigation.

6.3.2 Engineering Design Process Enhancement

As discussed in Chapter 3 little work has been done applying VR technology to the earliest stages of the Engineering Design Process. We proposed two new VR applications which may provide benefits in those early stages. First, during Opportunity Development, it can be difficult to understand and empathize with people who have different challenges than our own. However, gaining a thorough understanding of the challenges a particular product is trying to solve is an important part of the design process. Virtual Reality has been shown to increase empathy towards those with different challenges and hence we suggest a case study of using VR to aid in the development of empathy towards the target market for a particular product.

VR could also greatly aid in the process of developing novel solutions to design problems. As discussed in Section 3.4.2 exposure to a wide variety of creative stimuli during the design process can provide inspiration for new designs and solutions. Currently seeking out inspiration either requires travel or viewing of images on a 2D screen. VR has the potential to aid in this process by reducing the amount of travel required to experience the inspirational stimuli and allowing an immersive 3D viewing environment. In addition, in the virtual environment, artifacts and environments could be juxtaposed to create combinations not typically found in nature potentially inspiring additional creativity. A case study of VR for the purpose of creative inspiration is recommended to evaluate the benefits of such a system.

Finally, in many disciplines it is common to have users evaluate a proposed design during its development and provide feedback. In software related industries, it is common to do this early in the design process using low-fidelity prototypes [271]. This is not as common in engineering related design processes because of the resources required to create these prototypes. However, by augmenting the CVE presented in this work with additional drawing tools as suggested above, the CVE could be used as both a collaborative concept generation room where designers can quickly create and demonstrate 3D virtual prototypes as well as an environment where these low fidelity prototypes could be shared with end-user groups to get feedback early on in the design process to create more customer centric designs.

REFERENCES

[1] Kennedy, R. S., Lane, N. E., Berbaum, K. S., and Lilienthal, M. G., 1993. "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness." *The International Journal of Aviation Psychology,* **3**(3), pp. 203–220. vii, 72, 73

[2] Bouchard, S., Robillard, G., and Renaud, P., 2007. "Revising the factor structure of the simulator sickness questionnaire." *Annual review of cybertherapy and telemedicine,* **5**, pp. 128–137. vii, 73, 74

[3] Bouchard, S., St-Jacques, J., Renaud, P., and Wiederhold, B. K., 2009. "Side effects of immersions in virtual reality for people suffering from anxiety disorders." *Journal of CyberTherapy & Rehabilitation,* **2**(2). vii, 72, 73, 74

[4] Colgan, A., 2014. How does the leap motion controller work?, Aug `http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/`. viii, 34

[5] Kauff, P., and Schreer, O., 2002. "An immersive 3d video-conferencing system using shared virtual team user environments." In *Proceedings of the 4th international conference on Collaborative virtual environments*, ACM, pp. 105–112. 1, 78

[6] Huang, G. Q., 2002. "Web-based support for collaborative product design review." *Computers in Industry,* **48**(1), pp. 71–88. 1, 55

[7] Olson, G. M., and Olson, J. S., 2000. "Distance matters." *Human-computer interaction,* **15**(2), pp. 139–178. 1, 3, 7

[8] Olson, J. S., and Olson, G. M., 2014. "Bridging distance." In *Human-Computer Interaction and Management Information Systems: Applications. Advances in Management Information Systems*, V. Zwass, ed., pp. 101–118. 1, 3

[9] Bradner, E., and Mark, G., 2002. "Why distance matters: effects on cooperation, persuasion and deception." In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, p. 226235. 1

[10] Snowdon, D., Churchill, E. F., and Munro, A. J., 2001. "Collaborative virtual environments: Digital spaces and places for cscw: An introduction." In *Collaborative Virtual Spaces: Digital Plaecs and Spaces for Interaction*, Springer, pp. 3–17. 1, 2, 79

[11] Wood, J. T., 2012. *Interpersonal Communication: Everyday Encounters.*, 7th ed. Cengage Learning. 1

[12] Agrawala, M., Beers, A. C., McDowall, I., Fröhlich, B., Bolas, M., and Hanrahan, P., 1997. "The two-user responsive workbench: Support for collaboration through individual views of a shared space." In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 327–332. 1, 2, 4, 5, 6, 8, 24, 26, 55

[13] Greenberg, S., 2007. "Observing collaboration: Group-centered design." In *HCI Remixed: Reflections on Works that have influenced the HCI Community*, MIT Press, pp. 111–118. 1, 2, 4, 5

[14] Solomon, D., and Theiss, J., 2015. *Interpersonal Communication: Putting Theory into Practice*. Routledge, 711 Third Avenue, New York, NY 10017, ch. Nonverbal Communication, pp. 155–180. 2, 5

[15] Bowers, J., Pycock, J., and O'Brien, J., 1996. "Talk and embodiment in collaborative virtual environments." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, pp. 58–65. 2, 10

[16] Engelbart, D. C., 2001. "Augmenting human intellect: a conceptual framework (1962)." *PACKER, Randall and JORDAN, Ken. Multimedia. From Wagner to Virtual Reality. New York: WW Norton & Company*, pp. 64–90. 3

[17] , 2015. Nasa systems engineering processes and requirements, Apr. `http://nodis3.gsfc.nasa.gov/npg_img/n_pr_7123_001a_/n_pr_7123_001a_.pdf`. 3

[18] , 2015. Technical review definitions, Apr. `http://acc.dau.mil/CommunityBrowser.aspx?id=294561`. 3

[19] Tang, J. C., 1991. "Findings from observational studies of collaborative work." *International Journal of Man-machine studies,* **34**(2), pp. 143–160. 4, 5

[20] Benford, S., Greenhalgh, C., Rodden, T., and Pycock, J., 2001. "Collaborative virtual environments." *Commun. ACM,* **44**(7), July, pp. 79–85. 6, 24

[21] Grudin, J., 1994. "Groupware and social dynamics: Eight challenges for developers." *Communications of the ACM,* **37**(1), pp. 92–105. 6

[22] Palen, L., and Grudin, J., 2003. "Discretionary adoption of group support software: Lessons from calendar applications." In *Implementing collaboration technologies in industry*. Springer, pp. 159–180. 7

[23] , 2015. Google drive: Sharing and collaboration, July `http://www.gcflearnfree.org/googledriveanddocs/6`. 7

[24] , 2015. Office 365 collaboration tools, July `https://products.office.com/en-us/business/office-365-file-sharing-online-collaboration-tools`. 7

[25] , 2015. Autodesk a360: Collaboration in the cloud, July `http://www.autodesk.com/products/a360/overview`. 7

[26] Red, E., Holyoak, V., Jensen, C. G., Marshall, F., Ryskamp, J., and Xu, Y., 2010. "ν-cax: A research agenda for collaborative computer-aided applications." *Computer-Aided Design and Applications,* **7**(3), pp. 387–404. 7, 54, 55, 80

[27] Li, W., Lu, W., Fuh, J., and Wong, Y., 2005. "Collaborative computer-aided designresearch and development status." *Computer-Aided Design,* **37**(9), pp. 931–940. 7

[28] Mcgrath, A., and Prinz, W., 2001. "All that is solid melts into software." In *Collaborative Virtual Spaces: Digital Plaecs and Spaces for Interaction*, Springer, pp. 99–114. 7

[29] Bochenek, G. M., and Ragusa, J. M., 2004. "Improving integrated project team interaction through virtual (3d) collaboration." *Engineering Management Journal,* **16**(2), pp. 3–12. 8, 21, 24, 78, 80

[30] Satter, K., and Butler, A., 2015. "Competitive usability analysis of immersive virtual environments in engineering design review." *Journal of Computing and Information Science in Engineering,* **15**(3), Sep, pp. 031001–031001–12. 8, 24, 33, 80

[31] Banerjee, P., Bochenek, G. M., and Ragusa, J. M., 2002. "Analyzing the relationship of presence and immersive tendencies on the conceptual design review process." *Journal of Computing and Information Science in Engineering,* **2**(1), Jun, pp. 59–64. 8, 24

[32] Banerjee, P., and Basu-Mallick, D., 2003. "Measuring the effectiveness of presence and immersive tendencies on the conceptual design review process." *Journal of Computing and Information Science in Engineering,* **3**(2), Jun, pp. 166–169. 8, 24

[33] Starner, T., Mann, S., Rhodes, B., Levine, J., Healey, J., Kirsch, D., Picard, R. W., and Pentland, A., 1997. "Augmented reality through wearable computing." *Presence: Teleoperators and Virtual Environments,* **6**(4), Aug, pp. 386–398. 8, 22

[34] Sutherland, I. E., 1968. "A head-mounted three dimensional display." In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), ACM, pp. 757–764. 8, 21, 24, 25

[35] Jayaram, S., Vance, J., Gadh, R., Jayaram, U., and Srinivasan, H., 2001. "Assessment of vr technology and its applications to engineering problems." *Journal of Computing and Information Science in Engineering,* **1**(1), Jan, pp. 72–83. 8, 28, 33, 37

[36] Miller, S. A., Misch, N. J., and Dalton, A. J., 2005. "Low-Cost, Portable, Multi-Wall Virtual Reality." In *Eurographics Symposium on Virtual Environments*, E. Kjems and R. Blach, eds., The Eurographics Association. 8, 26, 54

[37] DeFanti, T. A., Acevedo, D., Ainsworth, R. A., Brown, M. D., Cutchin, S., Dawe, G., Doerr, K.-U., Johnson, A., Knox, C., Kooima, R., Kuester, F., Leigh, J., Long, L., Otto, P., Petrovic, V., Ponto, K., Prudhomme, A., Rao, R., Renambot, L., Sandin, D. J., Schulze, J. P., Smarr, L., Srinivasan, M., Weber, P., and Wickham, G., 2011. "The future of the cave." *Central European Journal of Engineering,* **1**(1), pp. 16–37. 9, 25, 32, 33

[38] Hsu, J., 2015. Virtual reality pioneer looks beyond entertainment, Apr. http://spectrum.ieee.org/tech-talk/consumer-electronics/portable-devices/virtual-reality-pioneer-looks-beyond-entertainment. 9

[39] Thompson III, F. V., 2014. "Evaluation of a commodity vr interaction device for gestural object manipulation in a three dimensional work environment." PhD thesis, Iowa State University. 9, 28, 35

[40] LaViola, Jr., J. J., 2000. "A discussion of cybersickness in virtual environments." *SIGCHI Bull.,* **32**(1), Jan., pp. 47–56. 9, 33, 56

[41] Rebenitsch, L., and Owen, C., 2014. "Individual variation in susceptibility to cybersickness." In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, ACM, pp. 309–317. 9, 33

[42] Huang, F.-C., Chen, K., and Wetzstein, G., 2015. "The light field stereoscope: Immersive computer graphics via factored near-eye light field displays with focus cues." *ACM Trans. Graph.,* **34**(4), July, pp. 60:1–60:12. 9, 21, 32

[43] Sabelman, E. E., and Lam, R., 2015. "The real-life dangers of augmented reality." *Spectrum, IEEE,* **52**(7), pp. 48–53. 9

[44] Greenhalgh, C., and Benford, S., 1995. "Massive: a distributed virtual reality system incorporating spatial trading." In *Proceedings of 15th International Conference on Distributed Computing Systems*, pp. 27–34. 10, 46

[45] Billinghurst, M., Poupyrev, I., Kato, H., and May, R., 2000. "Mixing realities in shared space: an augmented reality interface for collaborative computing." In *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, Vol. 3, pp. 1641–1644 vol.3. 10, 29

[46] Billinghurst, M., Weghorst, S., and Furness, T., 1998. "Shared space: An augmented reality approach for computer supported collaborative work." *Virtual Reality,* **3**(1), pp. 25–36. 10, 29

[47] Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pedersen, E., Pier, K., et al., 1992. "Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration." In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, pp. 599–607. 10

[48] Peña-Mora, F., Hussein, K., Vadhavkar, S., and Benjamin, K., 2000. "Cairo: a concurrent engineering meeting environment for virtual design teams." *Artificial Intelligence in Engineering,* **14**(3), pp. 203–219. 10

[49] Conti, G., Ucelli, G., and Petric, J., 2002. "Jcad-vr:: A collaborative design tool for architects." In *Proceedings of the 4th International Conference on Collaborative Virtual Environments*, CVE '02, ACM, pp. 153–154. 10, 24

[50] Smith, R. C., Pawlicki, R. R., Leigh, J., and Brown, D. A., 2000. "Collaborative visualeyes." *General Motors Research & Development Center*, pp. 1–8. 10

[51] , 2015. Lockheed collaborative human immersive laboratory, Aug. `http://www.lockheedmartin.com/us/products/chil.html`. 10

[52] , 2015. Informationweek showcases raytheon technology, Aug. `http://www.raytheon.com/news/feature/rms14_iweek_cave.html`. 10

[53] Dorozhkin, D. V., Vance, J. M., Rehn, G. D., and Lemessi, M., 2012. "Coupling of interactive manufacturing operations simulation and immersive virtual reality." *Virtual Reality,* **16**(1), pp. 15–23. 10, 24

[54] Behdad, S., Berg, L., Vance, J., and Thurston, D., 2014. "Immersive computing technology to investigate tradeoffs under uncertainty in disassembly sequence planning." *Journal of Mechanical Design,* **136**(7), p. 071001. 10, 79

[55] 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*, CVE '00, ACM, pp. 57–63. 10, 24, 26, 55, 78, 80

[56] Smith, R. C., 2001. "Shared vision." *Commun. ACM,* **44**(12), Dec., pp. 45–48. 13, 24, 28, 29, 36, 48

[57] Cruz-Neira, C., Sandin, D. J., and DeFanti, T. A., 1993. "Surround-screen projection-based virtual reality: The design and implementation of the cave." In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, ACM, pp. 135–142. 21, 25, 26, 32, 38

[58] Arthur, J. J., Bailey, R. E., Williams, S. P., Prinzel, L. J., Shelton, K. J., Jones, D. R., and Houston, V., 2015. "A review of head-worn display research at nasa langley research center." *Proc. SPIE,* **9470**, pp. 94700W–94700W–15. 21

[59] Davis, S., Nesbitt, K., and Nalivaiko, E., 2015. "Comparing the onset of cybersickness using the oculus rift and two virtual roller coasters." In *11th Australasian Conference on Interactive Entertainment (IE 2015)*, Y. Pisan, K. Nesbitt, and K. Blackmore, eds., Vol. 167 of *CRPIT*, ACS, pp. 3–14. 21, 24, 25, 33

[60] Steuer, J., 1992. "Defining virtual reality: Dimensions determining telepresence." *Journal of Communication,* **42**(4), pp. 73–93. 21, 22, 24

[61] Valin, S., Francu, A., Trefftz, H., and Marsic, I., 2001. "Sharing viewpoints in collaborative virtual environments." In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, pp. 12 pp.–. 21, 56

[62] Kress, B., and Starner, T., 2013. "A review of head-mounted displays (hmd) technologies and applications for consumer electronics." *Proc. SPIE,* **8720**, pp. 87200A–87200A–13. 23, 25

[63] Bordegoni, M., Cugini, U., Caruso, G., and Polistina, S., 2009. "Mixed prototyping for product assessment: a reference framework." *International Journal on Interactive Design and Manufacturing (IJIDeM),* **3**(3), pp. 177–187. 23, 49

[64] Ferrise, F., Graziosi, S., and Bordegoni, M., 2015. "Prototyping strategies for multisensory product experience engineering." *Journal of Intelligent Manufacturing*, pp. 1–13. 23

[65] Cecil, J., and Kanchanapiboon, A., 2007. "Virtual engineering approaches in product and process design." *The International Journal of Advanced Manufacturing Technology,* **31**(9), pp. 846–856. 24

[66] Cakmakci, O., and Rolland, J., 2006. "Head-worn displays: a review." *Journal of Display Technology,* **2**(3), Sept, pp. 199–216. 25

[67] OSVR, 2016. Open source headed-mounted display for osvr `http://www.osvr.org/hdk2.html`. 25, 27

[68] Creagh, H., 2003. "Cave automatic virtual environment." In *Proceedings: Electrical Insulation Conference and Electrical Manufacturing and Coil Winding Technology Conference (Cat. No.03CH37480)*, pp. 499–504. 25

[69] 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." *Commun. ACM,* **35**(6), June, pp. 64–72. 25, 26, 54

[70] Katz, A., 2015. Brown university unveils 3d virtual-reality room, June `https://www.bostonglobe.com/lifestyle/style/2015/06/19/brown-university-unveils-virtual-reality-room/QoTOOp66NpPZeGMF0bapjO/story.html`. 26, 54

[71] Ramsey, D., 2008. 3d virtual reality environment developed at uc san diego helps scientists innovate, Sept `http://www.calit2.net/newsroom/release.php?id=1383`. 26, 54

[72] Digital Trends Staff, 2016. Spec comparison: The rift is less expensive than the vive, but is it a better value? `http://www.digitaltrends.com/virtual-reality/oculus-rift-vs-htc-vive/`. 27

[73] Orland, K., 2016. The ars vr headset showdownoculus rift vs. htc vive `http://arstechnica.com/gaming/2016/04/the-ars-vr-headset-showdown-oculus-rift-vs-htc-vive/`. 27

[74] Avegant, 2016. Avegant glyph - product `https://www.avegant.com/product`. 27

[75] Avegant, 2015. Avegant glyph virtual reality headset `http://www.vrs.org.uk/virtual-reality-gear/head-mounted-displays/avegant-glyph.html`. 27

[76] Avegant, 2016. Avegant online store `https://shop.avegant.com/`. 27

[77] HTC, 2016. Vive — product hardware `https://www.htcvive.com/us/product/`. 27, 37

[78] Google, 2016. Google cardboard `https://vr.google.com/cardboard/`. 27

[79] Samsung, 2016. Samsung gear vr. 27

[80] Sony, 2016. Playstartion vr full specs. 27

[81] Bakalar, J., 2016. Sony playstation vr review, Oct https://www.cnet.com/products/sony-playstation-vr-review/. 27

[82] Dldolo, 2016. Dlodlo v1 http://www.dlodlo.com/en/v-one.html. 27, 31

[83] Krol, J., 2016. Dlodlo's v1 promises vr in a slim form factor, Aug https://www.cnet.com/products/dlodlo-glass-v1/preview/. 27

[84] FOVE, 2016. Fove http://www.getfove.com/. 27, 29, 31

[85] PLAFKE, J., 2015. Fove eye-tracking vr headset makes virtual reality more like reality, May. 27

[86] Starbreeze Studios, 2016. Starvr http://www.starvr.com/. 27, 31

[87] Martindale, J., 2015. Starvr specs make the oculus rift look like a kids toy, Jun http://www.digitaltrends.com/computing/starvr-vr-headset-5120-1440-resolution/. 27

[88] VRVANA, 2016. Vrvana - technology to augment our reality https://www.vrvana.com/. 27, 32

[89] Prasuethsut, L., 2015. Hands on: Sulon cortex review, Mar http://www.techradar.com/reviews/gaming/sulon-cortex-1288470/review. 27

[90] VRELIA, I., 2016. Go - immersion-vrelia http://immersionvrelia.com/go/. 27, 32

[91] VisusVR, 2016. Visusvr. 27, 32

[92] Labs, G., 2016. Gameface labs — the world's first wireless virtual reality head mounted console. 27, 32

[93] Tyrrel, B., 2015. Hands-on with gameface labs vr headset, jul http://www.ign.com/articles/2015/07/01/hands-on-with-gameface-labs-vr-headset. 27

[94] Desai, P. R., Desai, P. N., Ajmera, K. D., and Mehta, K., 2014. "A review paper on oculus rift-a virtual reality headset." *International Journal of Engineering Trends and Technology,* **13**(4), Jul, pp. 175–179. 28

[95] Goradia, I., Doshi, J., and Kurup, L., 2014. "A review paper on oculus rift & project morpheus." *International Journal of Current Engineering and Technology,* **4**(5), pp. 3196–3200. 28

[96] Mitroff, S., 2016. How to wear an oculus rift and htc vive with glasses http://www.cnet.com/how-to/how-to-wear-an-oculus-rift-and-htc-vive-with-glasses/. 28

[97] Mitchell, n., 2014. Palmer luckey and nate mitchell interview: Low persistence 'fundamentally changes the oculus rift experience' http://www.roadtovr.com/ces-2014-oculus-rift-crystal-cove-prototype-palmer-luckey-nate-mitchell-low-persistence-positional-tracking-interview-video/. 28

[98] Blog, O. R., 2016. John carmacks delivers some home truths on latency http://oculusrift-blog.com/john-carmacks-message-of-latency/682/. 28

[99] Li, H., Trutoiu, L., Olszewski, K., Wei, L., Trutna, T., Hsieh, P.-L., Nicholls, A., and Ma, C., 2015. "Facial performance sensing head-mounted display." *ACM Trans. Graph.,* **34**(4), July, pp. 47:1–47:9. 29, 82

[100] Matney, L., 2016. Veeso wants to share your smiles and eye-rolls in vr, Jul https://techcrunch.com/2016/07/25/veeso-wants-to-share-your-smiles-and-eye-rolls-in-vr/. 29

[101] Sony, 2016. Playstation vr https://www.playstation.com/en-us/explore/playstation-vr/. 31

[102] Dldolo, 2016. Dldolo glass h1 http://www.dlodlo.com/en/h-one.html. 31

[103] Technologies, S., 2016. Sulon http://www.sulon.com/. 32

[104] Akeley, K., Watt, S. J., Girshick, A. R., and Banks, M. S., 2004. "A stereo display prototype with multiple focal distances." *ACM Trans. Graph.,* **23**(3), Aug., pp. 804–813. 32

[105] Stanney, K. M., Kennedy, R. S., and Drexler, J. M., 1997. "Cybersickness is not simulator sickness." *Proceedings of the Human Factors and Ergonomics Society Annual Meeting,* **41**(2), pp. 1138–1142. 33

[106] Pavlik, R. A., and Vance, J. M., 2015. "Interacting with grasped objects in expanded haptic workspaces using the bubble technique." *Journal of Computing and Information Science in Engineering,* **15**(4), Oct, pp. 041006–041006–7. 33

[107] Deering, M. F., 1995. "Holosketch: A virtual reality sketching/animation tool." *ACM Trans. Comput.-Hum. Interact.,* **2**(3), Sept., pp. 220–238. 33

[108] Vélaz, Y., Rodríguez Arce, J., Gutiérrez, T., Lozano-Rodero, A., and Suescun, A., 2014. "The influence of interaction technology on the learning of assembly tasks using virtual reality." *Journal of Computing and Information Science in Engineering,* **14**(4), Oct, pp. 041007–041007–9. 33

[109] Holt, P., Ritchie, J., Day, P., Simmons, J., Robinson, G., Russell, G., and Ng, F., 2004. "Immersive virtual reality in cable and pipe routing: design metaphors and cognitive ergonomics." *Journal of Computing and Information Science in Engineering,* **4**(3), pp. 161–170. 33, 80

[110] Massie, T. H., and Salisbury, J. K., 1994. "The phantom haptic interface: A device for probing virtual objects." In *Proceedings of the ASME Dynamic Systems and Control Division*, pp. 295–301. 33

[111] Satter, K. M., and Butler, A. C., 2012. "Finding the value of immersive, virtual environments using competitive usability analysis." *Journal of Computing and Information Science in Engineering,* **12**(2), May, pp. 024504–024504–6. 33

[112] Sato, Y., Kobayashi, Y., and Koike, H., 2000. "Fast tracking of hands and fingertips in infrared images for augmented desk interface." In *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, pp. 462–467. 33

[113] Ribo, M., Pinz, A., and Fuhrmann, A. L., 2001. "A new optical tracking system for virtual and augmented reality applications." In *IMTC 2001. Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference. Rediscovering Measurement in the Age of Informatics (Cat. No.01CH 37188)*, Vol. 3, pp. 1932–1936 vol.3. 33

[114] Pintaric, T., and Kaufmann, H., 2007. "Affordable infrared-optical pose tracking for virtual and augmented reality." In *IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, G. Zachmann, ed., Shaker Verlag, pp. 44–51 Vortrag: IEEE Virtual Reality 2007, Charlotte, NC (USA); 2007-03-14 – 2007-03-17. 33

[115] Corazza, S., Mündermann, L., Chaudhari, A. M., Demattio, T., Cobelli, C., and Andriacchi, T. P., 2006. "A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach." *Annals of Biomedical Engineering,* **34**(6), pp. 1019–1029. 33

[116] Fitzgerald, D., Foody, J., Kelly, D., Ward, T., Markham, C., McDonald, J., and Caulfield, B., 2007. "Development of a wearable motion capture suit and virtual reality biofeedback system for the instruction and analysis of sports rehabilitation exercises." In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 4870–4874. 33

[117] Loy, G., Eriksson, M., Sullivan, J., and Carlsson, S., 2004. "Monocular 3d reconstruction of human motion in long action sequences." In *Computer Vision - ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part IV*, T. Pajdla and J. Matas, eds., Springer Berlin Heidelberg, pp. 442–455. 33

[118] Pullen, K., and Bregler, C., 2002. "Motion capture assisted animation: Texturing and synthesis." *ACM Trans. Graph.,* **21**(3), July, pp. 501–508. 33

[119] Edmonds, E., Turner, G., and Candy, L., 2004. "Approaches to interactive art systems." In *Proceedings of the 2Nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, GRAPHITE '04, ACM, pp. 113–117. 33

[120] Weichert, F., Bachmann, D., Rudak, B., and Fisseler, D., 2013. "Analysis of the accuracy and robustness of the leap motion controller." *Sensors,* **13**(5), May, p. 63806393. 35

[121] Guna, J., Jakus, G., Poganik, M., Tomai, S., and Sodnik, J., 2014. "An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking.." *Sensors (Basel, Switzerland),* **14**(2), pp. 3702–20. 35

[122] Microsoft, 2016. Kinect for windows sdk 1.8: Skeletal tracking `https://msdn.microsoft.com/en-us/library/hh973074.aspx`. 35

[123] Khoshelham, K., and Elberink, S. O., 2012. "Accuracy and resolution of kinect depth data for indoor mapping applications." *Sensors,* **12**(2), p. 1437. 35

[124] Dutta, T., 2012. "Evaluation of the kinect sensor for 3-d kinematic measurement in the workplace.." *Applied ergonomics,* **43**(4), Jul, pp. 645–9. 35

[125] Wang, Q., Kurillo, G., Ofli, F., and Bajcsy, R., 2015. "Evaluation of pose tracking accuracy in the first and second generations of microsoft kinect." In *Healthcare Informatics (ICHI), 2015 International Conference on*, pp. 380–389. 35

[126] Noitom, 2016. Perception neuron care & caution `https://neuronmocap.com/support/care-caution`. 36

[127] Hayward, V., Astley, O. R., Cruz-Hernandez, M., Grant, D., and Robles-De-La-Torre, G., 2004. "Haptic interfaces and devices." *Sensor Review,* **24**(1), pp. 16–29. 37

[128] Burdea, G. C., and Coiffet, P., 2003. *Virtual reality technology*., Vol. 1 John Wiley & Sons. 37

[129] Oculus, 2016. Oculus touch `https://www.oculus.com/en-us/touch/`. 37

[130] Volkov, S., and Vance, J. M., 2001. "Effectiveness of haptic sensation for the evaluation of virtual prototypes." *Journal of Computing and Information Science in Engineering,* **1**(2), Apr, pp. 123–128. 37

[131] Burdea, G. C., 1999. "Haptic feedback for virtual reality." In *Proceedings of the Virtual Reality and Prototyping Workshop*, Citeseer. 37

[132] Spanlang, B., Normand, J.-M., Giannopoulos, E., and Slater, M., 2010. "A first person avatar system with haptic feedback." In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, VRST '10, ACM, pp. 47–50. 37

[133] Kim, S., Hasegawa, S., Koike, Y., and Sato, M., 2002. "Tension based 7-dof force feedback device: Spidar-g." In *Proceedings IEEE Virtual Reality 2002*, pp. 283–284. 37

[134] Frisoli, A., Salsedo, F., Bergamasco, M., Rossi, B., and Carboncini, M. C., 2009. "A force-feedback exoskeleton for upper-limb rehabilitation in virtual reality." *Applied Bionics and Biomechanics,* **6**(2), pp. 115–126. 37

[135] Fisch, A., Mavroidis, C., Melli-Huber, J., and Bar-Cohen, Y., 2003. *Haptic Devices for VIrtual Reality, Telepresence, and Human-Assistive Robots*. The International Society for Optical Eningeering, ch. 4. 37

[136] Kruijff, E., Schmalstieg, D., and Beckhaus, S., 2006. "Using neuromuscular electrical stimulation for pseudo-haptic feedback." In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '06, ACM, pp. 316–319. 37

[137] Xia, P., 2016. "Haptics for product design and manufacturing simulation." *IEEE Transactions on Haptics,* **9**(3), pp. 358–375. 38

[138] Xia, P., Mendes Lopes, A., and Restivo, M. T., 2013. "A review of virtual reality and haptics for product assembly: from rigid parts to soft cables." *Assembly Automation,* **33**(2), pp. 157–164. 38

[139] Xia, P., Lopes, A. M., and Restivo, M. T., 2013. "A review of virtual reality and haptics for product assembly (part 1): Rigid parts." *Assembly Automation,* **33**(1), Feb, p. 6877. 38

[140] Burdea, G. C., 2000. "Haptics issues in virtual environments." In *Proceedings Computer Graphics International 2000*, pp. 295–302. 38

[141] Hartmann, W. M., 1999. "How we localize sound." *Physics today,* **52**, pp. 24–29. 38

[142] Brungart, D. S., 1998. "Near-field auditory localization." PhD thesis, Massachusetts Institute of Technology. 38

[143] Mller, H., Srensen, M. F., Hammershi, D., and Jensen, C. B., 1995. "Head-related transfer functions of human subjects." *J. Audio Eng. Soc,* **43**(5), pp. 300–321. 38

[144] Geronazzo, M., Spagnol, S., Bedin, A., and Avanzini, F., 2014. "Enhancing vertical localization with image-guided selection of non-individual head-related transfer functions." In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4463–4467. 38, 39

[145] Wenzel, E. M., Arruda, M., Kistler, D. J., and Wightman, F. L., 1993. "Localization using nonindividualized head-related transfer functions." *J. Acoust. Soc. Am.,* **94**(1), Jul, pp. 111–123. 38, 39

[146] Brungart, D. S., and Rabinowitz, W. M., 1999. "Auditory localization of nearby sources. Head-related transfer functions." *J. Acoust. Soc. Am.,* **106**(3 Pt 1), Sep, pp. 1465–1479. 39

[147] Zotkin, D. N., Duraiswami, R., and Davis, L. S., 2004. "Rendering localized spatial audio in a virtual auditory space." *Trans. Multi.,* **6**(4), Aug., pp. 553–564. 39, 82

[148] Zotkin, D. N., Duraiswami, R., Grassi, E., and Gumerov, N. A., 2006. "Fast head-related transfer function measurement via reciprocity.." *The Journal of the Acoustical Society of America,* **120**(4), Oct, pp. 2202–15. 39

[149] Lalwani, M., 2016. For vr to be truly immersive, it needs convincing sound to match `https://www.engadget.com/2016/01/22/vr-needs-3d-audio/`. 39, 82

[150] Heilig, M. L., 1962. Sensorama simulator, aug US Patent 3,050,870. 39

[151] Matsukura, H., Nihei, T., and Ishida, H., 2011. "Multi-sensorial field display: Presenting spatial distribution of airflow and odor." In *2011 IEEE Virtual Reality Conference*, pp. 119–122. 39

[152] Ariyakul, Y., and Nakamoto, T., 2011. "Olfactory display using a miniaturized pump and a saw atomizer for presenting low-volatile scents." In *2011 IEEE Virtual Reality Conference*, pp. 193–194. 39

[153] Bordegoni, M., and Carulli, M., 2016. "Evaluating industrial products in an innovative visual-olfactory environment." *Journal of Computing and Information Science in Engineering,* **16**(3), Jun, pp. 030904–030904–9. 39

[154] Miyaura, M., Narumi, T., Nishimura, K., Tanikawa, T., and Hirose, M., 2011. "Olfactory feedback system to improve the concentration level based on biological information." In *2011 IEEE Virtual Reality Conference*, pp. 139–142. 39

[155] Narumi, T., Kajinami, T., Nishizaka, S., Tanikawa, T., and Hirose, M., 2011. "Pseudo-gustatory display system based on cross-modal integration of vision, olfaction and gustation." In *2011 IEEE Virtual Reality Conference*, IEEE, pp. 127–130. 39

[156] Dieter, G. E., and Schmidt, L. C., 2013. *Engineering design.*, Vol. 3 McGraw-Hill New York. 40

[157] Haskins, C., Forsberg, K., Krueger, M., Walden, D., and Hamelin, D., 2006. "Systems engineering handbook." In *INCOSE,*. 40

[158] Royce, W. W., 1970. "Managing the development of large software systems." *Proceedings of IEEE WESCON,* **26**(8), pp. 328–338. 40

[159] Evans, J. H., 1959. "Basic design concepts." *Journal of the American Society for Naval Engineers,* **71**(4), pp. 671–678. 40

[160] Forsberg, K., and Mooz, H., 1991. "The relationship of system engineering to the project cycle." *INCOSE International Symposium,* **1**(1), pp. 57–65. 40

[161] Mooz, H., and Forsberg, K., 2001. "4.4.3 a visual explanation of development methods and strategies including the waterfall, spiral, vee, vee+, and vee++ models." *INCOSE International Symposium,* **11**(1), pp. 610–617. 40

[162] Sheard, S. A., 1996. "Twelve systems engineering roles." *INCOSE International Symposium,* **6**(1), pp. 478–485. 40

[163] Fritsch, J., Judice, A., Soini, K., and Tretten, P., 2009. "Storytelling and repetitive narratives for design empathy: Case suomenlinna." *Nordes*(2). 40

[164] Kouprie, M., and Visser, F. S., 2009. "A framework for empathy in design: stepping into and out of the user's life." *Journal of Engineering Design,* **20**(5), pp. 437–448. 40

[165] Slater, M., Spanlang, B., Sanchez-Vives, M. V., and Blanke, O., 2010. "First person experience of body transfer in virtual reality." *PLOS ONE,* **5**(5), 05, pp. 1–9. 41

[166] 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), pp. 779 – 787. 41

[167] Rosenberg, R. S., Baughman, S. L., and Bailenson, J. N., 2013. "Virtual superheroes: Using superpowers in virtual reality to encourage prosocial behavior." *PLOS ONE,* **8**(1), 01, pp. 1–9. 41

[168] Kilteni, K., Groten, R., and Slater, M., 2012. "The sense of embodiment in virtual reality." *Presence,* **21**(4), Nov, pp. 373–387. 41

[169] Kilteni, K., Normand, J.-M., Sanchez-Vives, M. V., and Slater, M., 2012. "Extending body space in immersive virtual reality: A very long arm illusion." *PLOS ONE,* **7**(7), 07, pp. 1–15. 42

[170] Normand, J.-M., Giannopoulos, E., Spanlang, B., and Slater, M., 2011. "Multisensory stimulation can induce an illusion of larger belly size in immersive virtual reality." *PLOS ONE,* **6**(1), 01, pp. 1–11. 42

[171] van der Hoort, B., Guterstam, A., and Ehrsson, H. H., 2011. "Being barbie: The size of ones own body determines the perceived size of the world." *PLOS ONE,* **6**(5), 05, pp. 1–10. 42

[172] Gonalves, M., Cardoso, C., and Badke-Schaub, P., 2014. "What inspires designers? preferences on inspirational approaches during idea generation." *Design Studies,* **35**(1), pp. 29 – 53. 42

[173] Ward, T., 1994. "Structured imagination: the role of category structure in exemplar generation." *Cognitive Psychology,* **27**(1), pp. 1 – 40. 42

[174] Chan, J., Dow, S. P., and Schunn, C. D., 2015. "Do the best design ideas (really) come from conceptually distant sources of inspiration?." *Design Studies,* **36**, pp. 31 – 58. 42

[175] Eckert, C., and Stacey, M., 1998. "Fortune favours only the prepared mind: Why sources of inspiration are essential for continuing creativity." *Creativity and Innovation Management,* **7**(1), pp. 9–16. 42

[176] Jones, K., Bradshaw, C., Papadopoulos, J., and Platzer, M., 2005. "Bio-inspired design of flapping-wing micro air vehicles." *Aeronautical Journal,* **109**(1098), pp. 385–394. 42

[177] Spolenak, R., Gorb, S., and Arzt, E., 2005. "Adhesion design maps for bio-inspired attachment systems." *Acta Biomaterialia,* **1**(1), pp. 5 – 13. 42

[178] Setchi, R., and Bouchard, C., 2010. "In search of design inspiration: A semantic-based approach." *Journal of Computing and Information Science in Engineering,* **10**(3), Sep, pp. 031006–031006–23. 42, 43

[179] Rieuf, V., Bouchard, C., and Aoussat, A., 2015. "Immersive moodboards, a comparative study of industrial design inspiration material." *Journal of Design Research,* **13**(1), pp. 78–106. 43

[180] Hsi, S., Linn, M. C., and Bell, J. E., 1997. "The role of spatial reasoning in engineering and the design of spatial instruction." *Journal of Engineering Education,* **86**(2), pp. 151–158. 43

[181] Bryson, S., 1993. "Virtual reality in scientific visualization." *Computers & Graphics,* **17**(6), pp. 679 – 685. 43, 46

[182] Purschke, F., Schulze, M., and Zimmermann, P., 1998. "Virtual reality-new methods for improving and accelerating the development process in vehicle styling and design." In *Proceedings. Computer Graphics International (Cat. No.98EX149)*, pp. 789–797. 43, 55

[183] Fiorentino, M., de Amicis, R., Monno, G., and Stork, A., 2002. "Spacedesign: A mixed reality workspace for aesthetic industrial design." In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, ISMAR '02, IEEE Computer Society, pp. 86–. 43

[184] Krause, F.-L., Göbel, M., Wesche, G., and Biahmou, T., 2004. "A three-stage conceptual design process using virtual environments." *WSCG '2004: Posters: The 12-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*. 43

[185] De Araùjo, B. R., Casiez, G., and Jorge, J. A., 2012. "Mockup builder: Direct 3d modeling on and above the surface in a continuous interaction space." In *Proceedings of Graphics Interface 2012*, GI '12, Canadian Information Processing Society, pp. 173–180. 44

[186] De Araújo, B. R., Casiez, G., Jorge, J. A., and Hachet, M., 2013. "Mockup builder: 3d modeling on and above the surface." *Computers & Graphics,* **37**(3), pp. 165–178. 44

[187] Bordegoni, M., 2010. "Haptic and sound interface for shape rendering." *Presence,* **19**(4), Aug, pp. 341–363. 44

[188] Kim, S., Berkley, J. J., and Sato, M., 2003. "A novel seven degree of freedom haptic device for engineering design." *Virtual Reality,* **6**(4), pp. 217–228. 44, 45

[189] Marks, S., Estevez, J. E., and Connor, A. M., 2016. "Towards the holodeck: Fully immersive virtual reality visualisation of scientific and engineering data." *CoRR,* **abs/1604.05797**. 44

[190] Connell, M., and Tullberg, O., 2002. "A framework for immersive {FEM} visualisation using transparent object communication in a distributed network environment." *Advances in Engineering Software,* **33**(710), pp. 453 – 459 Engineering Computational Technology & Computational Structures Technology. 44

[191] Bryson, S., and Levit, C., 1992. "The virtual wind tunnel." *IEEE Comput. Graph. Appl.,* **12**(4), July, pp. 25–34. 44

[192] Kuester, F., Bruckschen, R., Hamann, B., and Joy, K. I., 2001. "Visualization of particle traces in virtual environments." In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '01, ACM, pp. 151–157. 44

[193] Bruno, F., Caruso, F., nDe Napoli, L., and Muzzupappa, M., 2006. "Visualization of industrial engineering data in augmented reality." *Journal of Visualization,* **9**(3), pp. 319–329. 44

[194] Fiorentino, M., Monno, G., and Uva, A., 2009. "Interactive touch and see fem simulation using augmented reality." *International Journal of Engineering Education,* **25**(6). 45

[195] Uva, A., Cristiano, S., Fiorentino, M., and Monno, G., 2010. "Distributed design review using tangible augmented technical drawings." *Computer-Aided Design,* **42**(5), pp. 364 – 372 Advanced and Emerging Virtual and Augmented Reality Technologies in Product Design. 45, 47

[196] Lee, E.-J., and El-Tawil, S., 2008. "Femvrml: An interactive virtual environment for visualization of finite element simulation results." *Advances in Engineering Software,* **39**(9), pp. 737 – 742. 45

[197] Hambli, R., Chamekh, A., and Salah, H. B. H., 2006. "Real-time deformation of structure using finite element and neural networks in virtual reality applications." *Finite Elements in Analysis and Design,* **42**(11), pp. 985 – 991. 45

[198] Ryken, M. J., and Vance, J. M., 2000. "Applying virtual reality techniques to the interactive stress analysis of a tractor lift arm." *Finite Elem. Anal. Des.,* **35**(2), May, pp. 141–155. 45

[199] Lundin, K. E., Sillen, M., Cooper, M. D., and Ynnerman, A., 2005. "Haptic visualization of computational fluid dynamics data using reactive forces." *Proc. SPIE,* **5669**, pp. 31–41. 45

[200] Lawrence, D. A., Lee, C. D., Pao, L. Y., and Novoselov, R. Y., 2000. "Shock and vortex visualization using a combined visual/haptic interface." In *Proceedings Visualization 2000. VIS 2000 (Cat. No.00CH37145)*, pp. 131–137. 45

[201] Ferrise, F., Bordegoni, M., Marseglia, L., Fiorentino, M., and Uva, A. E., 2015. "Can interactive finite element analysis improve the learning of mechanical behavior of materials? a case study." *Computer-Aided Design and Applications,* **12**(1), pp. 45–51. 45

[202] Ferrise, F., Bordegoni, M., Fiorentino, M., and Uva, A. E., 2013. "Integration of realtime finite element analysis and haptic feedback for hands-on learning of the mechanical behavior of materials." In *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, p. V02BT02A037. 45

[203] Ware, C., and Franck, G., 1996. "Evaluating stereo and motion cues for visualizing information nets in three dimensions." *ACM Trans. Graph.,* **15**(2), Apr., pp. 121–140. 45

[204] Nelson, L., Cook, D., and Cruz-Neira, C., 1999. "Xgobi vs the c2: Results of an experiment comparing data visualization in a 3-d immersive virtual reality environment with a 2-d workstation display." *Computational Statistics,* **14**(1), pp. 39–51. 46

[205] Donalek, C., Djorgovski, S. G., Cioc, A., Wang, A., Zhang, J., Lawler, E., Yeh, S., Mahabal, A., Graham, M., Drake, A., Davidoff, S., Norris, J. S., and Longo, G., 2014. "Immersive and collaborative data visualization using virtual reality platforms." In *2014 IEEE International Conference on Big Data (Big Data)*, pp. 609–614. 46

[206] Bryson, S., and Levit, C., 1991. "The virtual windtunnel-an environment for the exploration of three-dimensional unsteady flows." In *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, pp. 17–24, 407. 46

[207] Brooks, Jr., F. P., Ouh-Young, M., Batter, J. J., and Jerome Kilpatrick, P., 1990. "Project gropehaptic displays for scientific visualization." *SIGGRAPH Comput. Graph.,* **24**(4), Sept., pp. 177–185. 46

[208] Santos, P., Stork, A., Gierlinger, T., Pagani, A., Paloc, C., Barandarian, I., Conti, G., de Amicis, R., Witzel, M., Machui, O., Jiménez, J. M., Araujo, B., Jorge, J., and Bodammer, G., 2007. "Improve: An innovative application for collaborative mobile mixed reality design

review." *International Journal on Interactive Design and Manufacturing (IJIDeM),* **1**(2), pp. 115–126. 46, 47

[209] Churchill, E. F., and Snowdon, D., 1998. "Collaborative virtual environments: An introductory review of issues and systems." *Virtual Reality,* **3**(1), pp. 3–15. 46

[210] Chryssolouris, G., Mavrikios, D., Pappas, M., Xanthakis, E., and Smparounis, K., 2009. *A Web and Virtual Reality-based Platform for Collaborative Product Review and Customisation*. Springer London, London, pp. 137–152. 46

[211] Hren, G., and Jezernik, A., 2009. "A framework for collaborative product review." *The International Journal of Advanced Manufacturing Technology,* **42**(7), pp. 822–830. 46

[212] Greenhalgh, C. M., 1997. "Evaluating the network and usability characteristics of virtual reality conferencing." *BT Technology Journal,* **15**(4), pp. 101–119. 46

[213] Lehner, V. D., and DeFanti, T. A., 1997. "Distributed virtual reality: supporting remote collaboration in vehicle design." *IEEE Computer Graphics and Applications,* **17**(2), Mar, pp. 13–17. 46

[214] Kan, H., Duffy, V. G., and Su, C.-J., 2001. "An internet virtual reality collaborative environment for effective product design." *Computers in Industry,* **45**(2), pp. 197 – 213. 46

[215] Szalavári, Z., Schmalstieg, D., Fuhrmann, A., and Gervautz, M., 1998. ""studierstube": An environment for collaboration in augmented reality." *Virtual Reality,* **3**(1), pp. 37–48. 47

[216] Castronovo, F., Nikolic, D., Liu, Y., and Messner, J., 2013. "An evaluation of immersive virtual reality systems for design reviews." In *Proceedings of the 13th International Conference on Construction Applications of Virtual Reality*. 47

[217] Zorriassatine, F., Wykes, C., Parkin, R., and Gindy, N., 2003. "A survey of virtual prototyping techniques for mechanical product development." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture,* **217**(4), pp. 513–530. 47

[218] Wang, G. G., 2003. "Definition and review of virtual prototyping." *Journal of Computing and Information Science in Engineering,* **2**(3), Jan, pp. 232–236. 47

[219] Ferrise, F., Bordegoni, M., and Cugini, U., 2013. "Interactive virtual prototypes for testing the interaction with new products." *Computer-Aided Design and Applications,* **10**(3), pp. 515–525. 48

[220] Wickman, C., and Sderberg, R., 2003. "Increased concurrency between industrial and engineering design using cat technology combined with virtual reality." *Concurrent Engineering,* **11**(1), pp. 7–15. 48

[221] Bordegoni, M., Colombo, G., and Formentini, L., 2006. "Haptic technologies for the conceptual and validation phases of product design." *Computers & Graphics,* **30**(3), pp. 377 – 390. 48

[222] Bordegoni, M., and Ferrise, F., 2013. "Designing interaction with consumer products in a multisensory virtual reality environment." *Virtual and Physical Prototyping,* **8**(1), pp. 51–64. 48

[223] Bruno, F., and Muzzupappa, M., 2010. "Product interface design: A participatory approach based on virtual reality." *International Journal of Human-Computer Studies,* **68**(5), pp. 254 – 269. 49

[224] Kanai, S., Horiuchi, S., Kikuta, Y., Yokoyama, A., and Shiroma, Y., 2007. "An integrated environment for testing and assessing the usability of information appliances using digital and physical mock-ups." In *Virtual Reality: Second International Conference, ICVR 2007, Held as part of HCI International 2007, Beijing, China, July 22-27, 2007. Proceedings*, R. Shumaker, ed., Springer Berlin Heidelberg, pp. 478–487. 49

[225] Kimishima, Y., and Aoyama, H., 2006. "Evaluation method of style design by using mixed reality technology." *Proceedings of JSPEThe Japan Society for Precision Engineering*. 49

[226] Verlinden, J., and Horváth, I., 2006. "Framework for testing and validating interactive augmented prototyping as a design means in industrial practice." In *Proceedings of Virtual Concept*, pp. 1–10. 49

[227] Tideman, M., van der Voort, M. C., and van Houten, F. J. A. M., 2008. "A new product design method based on virtual reality, gaming and scenarios." *International Journal on Interactive Design and Manufacturing (IJIDeM),* **2**(4), pp. 195–205. 49

[228] Nam, T.-J., and Lee, W., 2003. "Integrating hardware and software: Augmented reality based prototyping method for digital products." In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, ACM, pp. 956–957. 49

[229] Noon, C., Zhang, R., Winer, E., Oliver, J., Gilmore, B., and Duncan, J., 2012. "A system for rapid creation and assessment of conceptual large vehicle designs using immersive virtual reality." *Computers in Industry,* **63**(5), pp. 500 – 512. 49

[230] Kuehne, R., and Oliver, J., 1995. "A virtual environment for interactive assembly planning and evaluation." In *Proceedings of the ASME Design Automation Conference*. 50

[231] Xiaoling, W., Peng, Z., Zhifang, W., Yan, S., Bin, L., and Yangchun, L., 2004. "Development an interactive vr training for cnc machining." In *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '04, ACM, pp. 131–133. 50

[232] Angster, S. R., and Jayaram, S., 1996. "Vedam: virtual environments for design and manufacturing." PhD thesis, Washington State University Pullman, WA. 50

[233] Lim, T., Thin, A., Ritchie, J., Sung, R., Liu, Y., and Kosmadoudi, Z., 2010. *Haptic virtual reality assembly-moving towards real engineering applications*. INTECH Open Access Publisher. 50

[234] Coutee, A. S., McDermott, S. D., and Bras, B., 2001. "A haptic assembly and disassembly simulation environment and associated computational load optimization techniques." *Journal of Computing and Information Science in Engineering,* **1**(2), Jun, pp. 113–122. 50

[235] Coutee, A. S., and Bras, B., 2002. "Collision detection for virtual objects in a haptic assembly and disassembly simulation environment." In *ASME 2002 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, ASME, pp. 11–20. 50

[236] Seth, A., Vance, J. M., and Oliver, J. H., 2011. "Virtual reality for assembly methods prototyping: a review." *Virtual Reality,* **15**(1), pp. 5–20. 50

[237] Choi, S., Jung, K., and Noh, S. D., 2015. "Virtual reality applications in manufacturing industries: Past research, present findings, and future directions." *Concurrent Engineering,* **23**(1), pp. 40–63. 50

[238] Eubanks, C. F., and Ishii, K., 1993. "Ai methods for life-cycle serviceability design of mechanical systems." *Artificial Intelligence in Engineering,* **8**(2), pp. 127 – 140. 50

[239] Ishii, K., Eubanks, C., and Marks, M., 1993. "Evaluation methodology for post-manufacturing issues in life-cycle design." *Concurrent Engineering,* **1**(1), pp. 61–68. 50

[240] Gynn, M., and Steele, J., 2015. "Virtual automotive maintenance and service confirmation." In *SAE Technical Paper*, SAE International. 50

[241] de S, A. G., and Zachmann, G., 1999. "Virtual reality as a tool for verification of assembly and maintenance processes." *Computers & Graphics,* **23**(3), pp. 389 – 403. 51

[242] Borro, D., Savall, J., Amundarain, A., Gil, J. J., Garcia-Alonso, A., and Matey, L., 2004. "A large haptic device for aircraft engine maintainability." *IEEE Computer Graphics and Applications,* **24**(6), Nov, pp. 70–74. 51

[243] Peng, G., Hou, X., Gao, J., and Cheng, D., 2012. "A visualization system for integrating maintainability design and evaluation at product design stage." *The International Journal of Advanced Manufacturing Technology,* **61**(1), pp. 269–284. 51

[244] Coburn, J. Q., Freeman, I. J., and Salmon, J. L., in press. "Capabilities of current generation virtual reality to enhance the design process." *Journal of Computing and Information Science in Engineering*. 54, 78, 81

[245] Moghadam, K. R., and Ragan, E. D., 2017. "Towards understanding scene transition techniques in immersive 360 movies and cinematic experiences." In *2017 IEEE Virtual Reality (VR)*, pp. 375–376. 54

[246] Barbosa, C., Feijó, B., Dreux, M., Melo, R., and Scheer, S., 2003. "Distributed object model for collaborative cad environments based on design history." *Advances in Engineering Software,* **34**(10), pp. 621 – 631. 55

[247] Chen, X., Fuh, J., Wong, Y., Lu, Y., Li, W., and Qiu, Z., 2005. "An adaptable model for distributed collaborative design." *Computer-Aided Design and Applications,* **2**(1-4), pp. 47–55. 55

[248] Red, E., Jensen, G., French, D., and Weerakoon, P., 2011. "Multi-user architectures for computer-aided engineering collaboration." In *2011 17th International Conference on Concurrent Enterprising*, pp. 1–10. 55

[249] Nysetvold, T., and Teng, C. C., 2013. "Collaboration tools for multi-user cad." In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 241–245. 55

[250] Berg, L. P., and Vance, J. M., 2017. "An industry case study: Investigating early design decision making in virtual reality." *Journal of Computing and Information Science in Engineering,* **17**(1), p. 011001. 55, 79, 80

[251] Hindmarsh, J., Fraser, M., Heath, C., Benford, S., and Greenhalgh, C., 1998. "Fragmented interaction: Establishing mutual orientation in virtual environments." In *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, CSCW '98, ACM, pp. 217–226. 56

[252] Frees, S., 2010. "Context-driven interaction in immersive virtual environments." *Virtual Reality,* **14**(4), pp. 277–290. 56

[253] Chellali, A., Milleville-Pennel, I., and Dumas, C., 2013. "Influence of contextual objects on spatial interactions and viewpoints sharing in virtual environments." *Virtual Reality,* **17**(1), pp. 1–15. 56, 79

[254] So, R. H., Lo, W. T., and Ho, A. T., 2001. "Effects of navigation speed on motion sickness caused by an immersive virtual environment.." *Human factors,* **43**(3), pp. 452–61. 56

[255] Bowman, D. A., Koller, D., and Hodges, L. F., 1997. "Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques." In *Proceedings of the 1997 Virtual Reality Annual International Symposium (VRAIS '97)*, VRAIS '97, IEEE Computer Society, pp. 45–. 56, 57, 74

[256] Oculus, 2017. Oculus best practices version 310-30000-02 Accessed 1 March 2017. 57, 66, 75

[257] Elvezio, C., Sukan, M., Feiner, S., and Tversky, B., 2017. "Travel in large-scale head-worn vr: Pre-oriented teleportation with wims and previews." In *2017 IEEE Virtual Reality (VR)*, pp. 475–476. 57, 105

[258] Nguyen, T. T. H., Duval, T., and Fleury, C., 2013. "Guiding Techniques for Collaborative Exploration in Multi-Scale Shared Virtual Environments." In *GRAPP International Conference on Computer Graphics Theory and Applications*, pp. 327–336. 75

[259] Rosenberg, M., and Vance, J. M., 2014. "Investigating the use of large-scale immersive computing environments in collaborative design." In *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. V003T04A018–V003T04A018. 78

[260] Onosato, M., and Iwata, K., 1993. "Development of a virtual manufacturing system by integrating product models and factory models." *CIRP Annals - Manufacturing Technology,* **42**(1), pp. 475 – 478. 78

[261] Behdad, S., Berg, L. P., Thurston, D., and Vance, J., 2014. "Leveraging virtual reality experiences with mixed-integer nonlinear programming visualization of disassembly sequence planning under uncertainty." *Journal of Mechanical Design,* **136**(4), p. 041005. 79

[262] Seth, A., Vance, J. M., and Oliver, J. H., 2010. "Combining dynamic modeling with geometric constraint management to support low clearance virtual manual assembly." *Journal of Mechanical Design,* **132**(8), p. 081002. 79

[263] Magga, O. H., 2006. "Diversity in saami terminology for reindeer, snow, and ice." *International Social Science Journal,* **58**(187), pp. 25–34. 80

[264] Krupnik, I., and Müller-Wille, L., 2010. *Franz Boas and Inuktitut Terminology for Ice and Snow: From the Emergence of the Field to the "Great Eskimo Vocabulary Hoax"*. Springer Netherlands, Dordrecht, ch. 16, pp. 377–400. 80

[265] Wobbrock, J. O., Wilson, A. D., and Li, Y., 2007. "Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes." In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, ACM, pp. 159–168. 92

[266] Anthony, L., and Wobbrock, J. O., 2010. "A lightweight multistroke recognizer for user interface prototypes." In *Proceedings of Graphics Interface 2010*, GI '10, Canadian Information Processing Society, pp. 245–252. 92

[267] Vatavu, R.-D., Anthony, L., and Wobbrock, J. O., 2012. "Gestures as point clouds: A $p recognizer for user interface prototypes." In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, ACM, pp. 273–280. 92

[268] Freeman, I. J., Coburn, J. Q., and Salmon, J. L., in press. "Ca bi-directional interface for improved interaction with engineering models in virtual reality design reviews international journal on interactive design and manufacturing." *International Journal on Interactive Design and Manufacturing*. 105

[269] Knödel, S., Hachet, M., and Guitton, P., 2008. "Navidget for immersive virtual environments." In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, ACM, pp. 47–50. 105

[270] Tan, D. S., Robertson, G. G., and Czerwinski, M., 2001. "Exploring 3d navigation: Combining speed-coupled flying with orbiting." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '01, ACM, pp. 418–425. 105

[271] Hussain, Z., Slany, W., and Holzinger, A., 2009. *Current State of Agile User-Centered Design: A Survey*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 416–427. 106

APPENDIX A. POST EXPERIMENTAL SURVEY FOR SHARED CONTEXT CREATION STYLE STUDY

View Transition Survey

Q1 Enter your Subject Number

Q15 Movement Set 1

	None (1)	Slight (2)	Moderate (3)	Severe (4)
General Discomfort (1)	❍	❍	❍	❍
Fatigue (2)	❍	❍	❍	❍
Headache (3)	❍	❍	❍	❍
Eye Strain (4)	❍	❍	❍	❍
Difficulty Focusing (5)	❍	❍	❍	❍
Salivation Increase (6)	❍	❍	❍	❍
Sweating (7)	❍	❍	❍	❍
Nausea (8)	❍	❍	❍	❍
Difficulty Concentrating (9)	❍	❍	❍	❍
Blurred Vision (10)	❍	❍	❍	❍
Dizziness with eyes open (11)	❍	❍	❍	❍
Dizziness with eyes closed (12)	❍	❍	❍	❍
Vertigo (13)	❍	❍	❍	❍
Stomach Awareness (14)	❍	❍	❍	❍
Burping (15)	❍	❍	❍	❍

Q17 Movement Set 2

	None (1)	Slight (2)	Moderate (3)	Severe (4)
General Discomfort (1)	❍	❍	❍	❍
Fatigue (2)	❍	❍	❍	❍
Headache (3)	❍	❍	❍	❍
Eye Strain (4)	❍	❍	❍	❍
Difficulty Focusing (5)	❍	❍	❍	❍
Salivation Increase (6)	❍	❍	❍	❍
Sweating (7)	❍	❍	❍	❍
Nausea (8)	❍	❍	❍	❍
Difficulty Concentrating (9)	❍	❍	❍	❍
Blurred Vision (10)	❍	❍	❍	❍
Dizziness with eyes open (11)	❍	❍	❍	❍
Dizziness with eyes closed (12)	❍	❍	❍	❍
Vertigo (13)	❍	❍	❍	❍
Stomach Awareness (14)	❍	❍	❍	❍
Burping (15)	❍	❍	❍	❍

Q18 Movement Set 3

	None (1)	Slight (2)	Moderate (3)	Severe (4)
General Discomfort (1)	❍	❍	❍	❍
Fatigue (2)	❍	❍	❍	❍
Headache (3)	❍	❍	❍	❍
Eye Strain (4)	❍	❍	❍	❍
Difficulty Focusing (5)	❍	❍	❍	❍
Salivation Increase (6)	❍	❍	❍	❍
Sweating (7)	❍	❍	❍	❍
Nausea (8)	❍	❍	❍	❍
Difficulty Concentrating (9)	❍	❍	❍	❍
Blurred Vision (10)	❍	❍	❍	❍
Dizziness with eyes open (11)	❍	❍	❍	❍
Dizziness with eyes closed (12)	❍	❍	❍	❍
Vertigo (13)	❍	❍	❍	❍
Stomach Awareness (14)	❍	❍	❍	❍
Burping (15)	❍	❍	❍	❍

Q19 Movement Set 4

	None (1)	Slight (2)	Moderate (3)	Severe (4)
General Discomfort (1)	❍	❍	❍	❍
Fatigue (2)	❍	❍	❍	❍
Headache (3)	❍	❍	❍	❍
Eye Strain (4)	❍	❍	❍	❍
Difficulty Focusing (5)	❍	❍	❍	❍
Salivation Increase (6)	❍	❍	❍	❍
Sweating (7)	❍	❍	❍	❍
Nausea (8)	❍	❍	❍	❍
Difficulty Concentrating (9)	❍	❍	❍	❍
Blurred Vision (10)	❍	❍	❍	❍
Dizziness with eyes open (11)	❍	❍	❍	❍
Dizziness with eyes closed (12)	❍	❍	❍	❍
Vertigo (13)	❍	❍	❍	❍
Stomach Awareness (14)	❍	❍	❍	❍
Burping (15)	❍	❍	❍	❍

Q20 Stop Here The following questions are for the participant.

Q2 Please Rank you movement style preference for the task performed, with 1 being your most preferred style and 4 being your least preferred style.

_____ Teleport (1)
_____ Fade-out/Fade-in (2)
_____ Fly Around (3)
_____ Manual Movement (4)

Q3 Compared to "Teleport", how much physical discomfort (cybersickness) did you experience with the following movement styles?

	A lot less than Teleport (1)	Slightly less than Teleport (2)	About the same as Teleport (3)	Slightly more than Teleport (4)	Much more than Teleport (5)
Fade-out/Fade-in (1)	❍	❍	❍	❍	❍
Fly Around (2)	❍	❍	❍	❍	❍
Manual (3)	❍	❍	❍	❍	❍

Q5 In general, how lost did you feel after immediately following a movement?

	Knew exactly where I was (1)	A little lost (2)	Somewhat lost (3)	Very lost (4)	Totally lost (5)
Teleport (1)	❍	❍	❍	❍	❍
Fade-out/Fade-in (2)	❍	❍	❍	❍	❍
Fly Around (3)	❍	❍	❍	❍	❍
Manual (4)	❍	❍	❍	❍	❍

Q7 During the actual movement, how disorienting were the following movement styles?

	Not disorienting at all (1)	A little disorienting (2)	Moderately disorienting (3)	Very disorienting (4)	Extremely disorienting (5)
Teleport (1)	❍	❍	❍	❍	❍
Fade-out/Fade-in (2)	❍	❍	❍	❍	❍
Fly Around (3)	❍	❍	❍	❍	❍
Manual (4)	❍	❍	❍	❍	❍

Q12 Compared to "Teleportation", how long did it take for you to regain your bearings after a movement of the following styles?

	Never lost bearings (1)	Much less than Teleport (2)	Slightly less than Teleport (3)	About the same as Teleport (4)	Slight longer than Teleport (5)	Much longer than Teleport (6)
Fade-out/Fade-in (1)	❍	❍	❍	❍	❍	❍
Fly Around (2)	❍	❍	❍	❍	❍	❍
Manual (3)	❍	❍	❍	❍	❍	❍

Q8 Why is "${q://QID2/ChoiceGroup/ChoiceWithLowestValue}" your preferred movement style for the task?

Q9 Why is "${q://QID2/ChoiceGroup/ChoiceWithHighestValue}" your least preferred movement style for the task?

Q10 What did you like or dislike (if anything) about the other movement styles?

Q11 Would any of these movement styles be showstoppers (i.e. bad enough to cause you to avoid using the tool) if they were implemented in a VR tool?

- ❑ Teleport (1)
- ❑ Fade-in/Fade-out (2)
- ❑ Fly (3)
- ❑ Manual (4)
- ❑ None of the above (5)

Q11 Given the movement styles you experienced, is there a different movement style you would believe you might prefer for the task? If so please describe it.

Q6 Please list how much experience you have with the following items in your lifetime.

	0-5 (1)	5-10 (2)	10-30 (3)	30-100 (4)	>100 (5)
Hour of Virtual Reality (1)	❍	❍	❍	❍	❍
Hours playing Video games (2)	❍	❍	❍	❍	❍
Hours watching 3D Movies/TV (3)	❍	❍	❍	❍	❍
Hours watching Non-3D Movies/TV (4)	❍	❍	❍	❍	❍
Hours riding in a car (5)	❍	❍	❍	❍	❍
Rides on a roller-coaster (6)	❍	❍	❍	❍	❍
Rides on a swinging boat style amusement ride (7)	❍	❍	❍	❍	❍
Rides on a spinning style amusement ride (8)	❍	❍	❍	❍	❍

Q20 In general, how prone to Motion Sickness would you rate yourself?

- ❍ Never get Motion Sickness (1)
- ❍ Rarely/Only in Extreme Conditions (2)
- ❍ Experience Occasional Motion Sickness (3)
- ❍ Occasionally avoid certain activities because of Motion Sickness (4)
- ❍ Sometimes avoid certain activities because of Motion Sickness (5)
- ❍ Completely avoid certain activities because of Motion Sickness (6)

Q22 Do you have any other feedback you would like to share?

APPENDIX B. POST EXPERIMENTAL SURVEY RESULTS FOR SHARED CONTEXT CREATION STYLE

Default Report

View Transition Survey

June 3rd 2017, 10:44 pm MDT

Q1 - Enter your Subject Number

Enter your Subject Number
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29

16

26

28

27

25

24

23

22

21

20

19

18

17

15

14

13

12

11

10

9

8

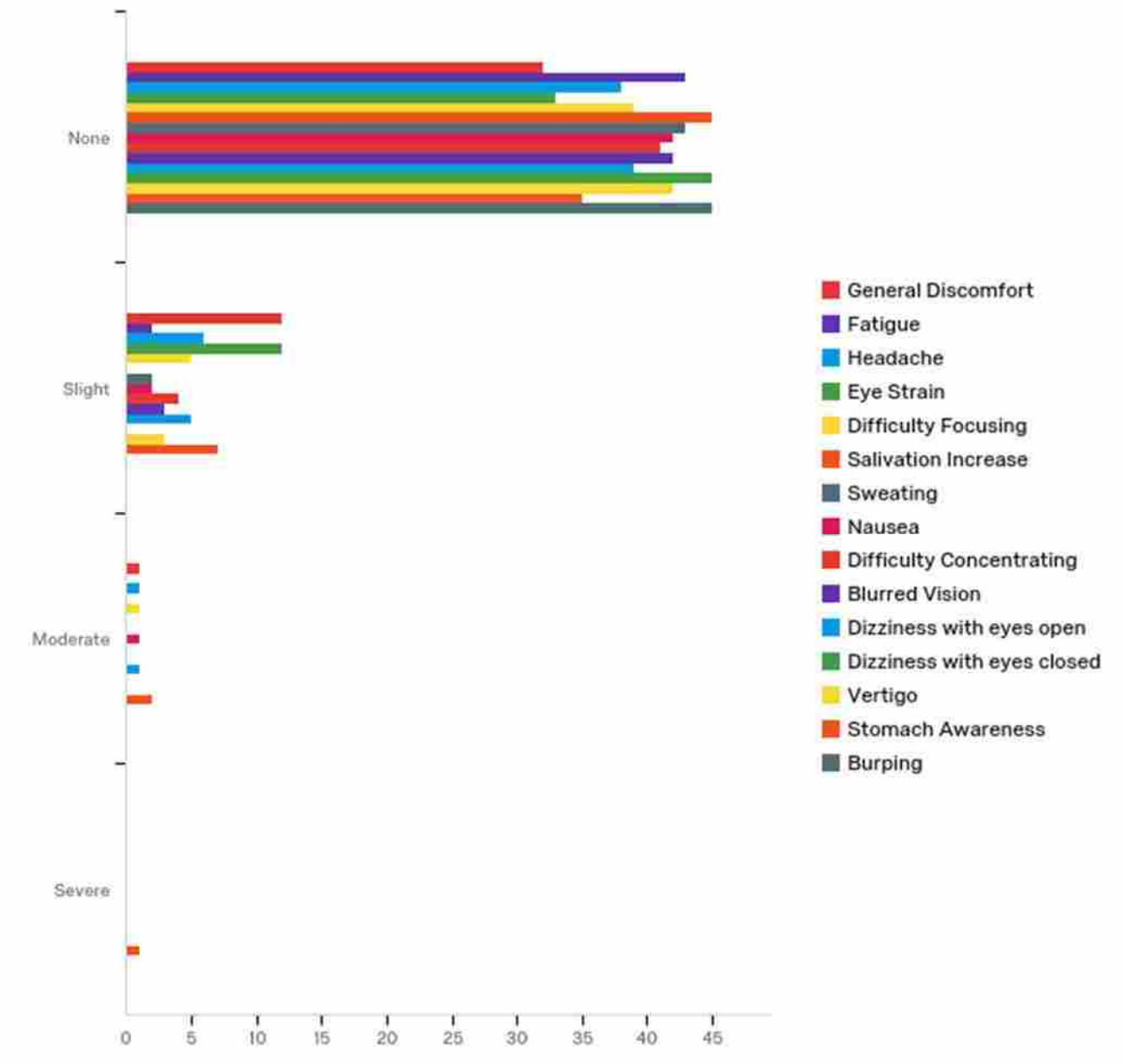
7

6

5

4

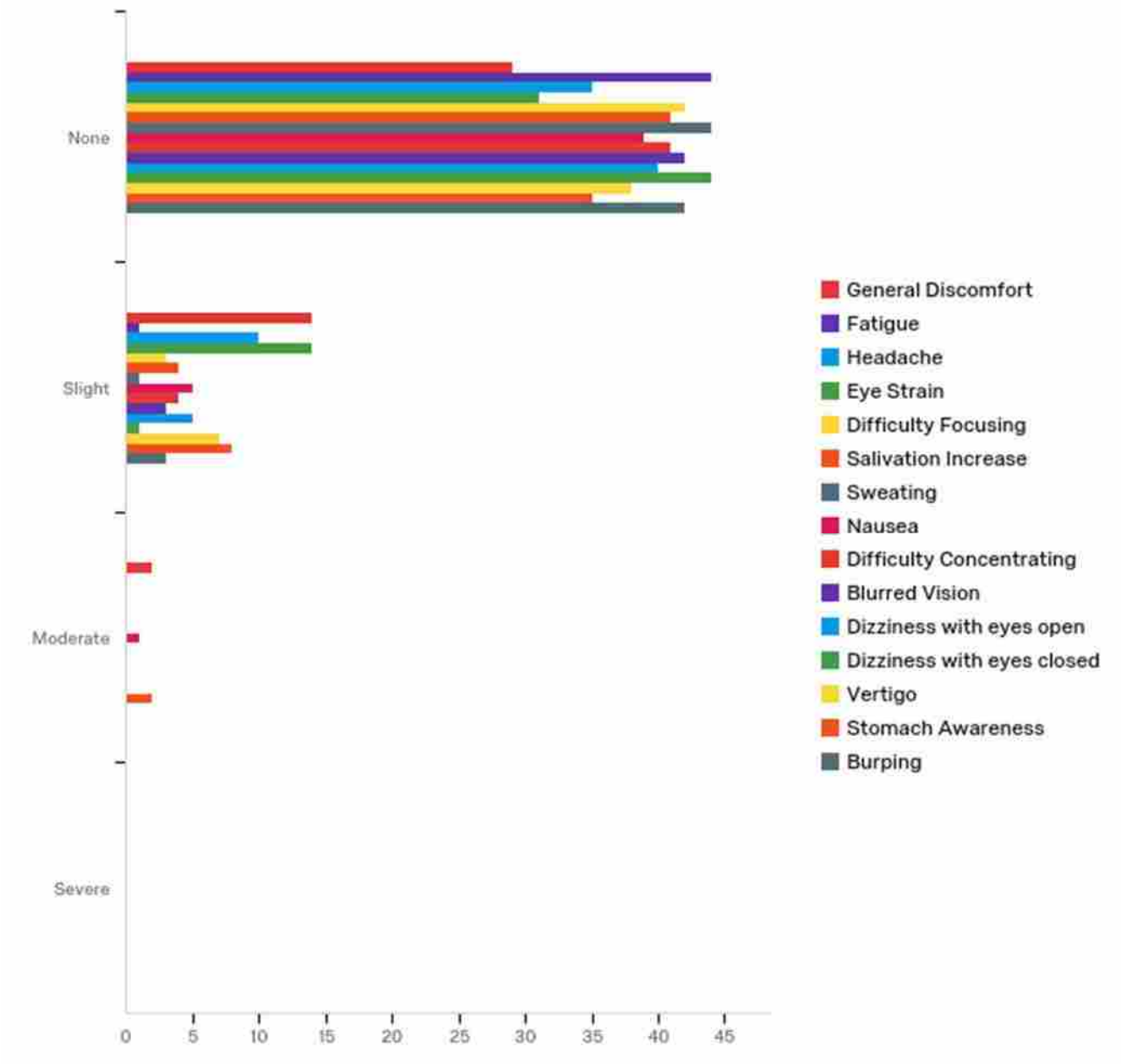
Q15 - Movement Set 1



#	Question	None		Slight		Moderate		Severe		Total
1	General Discomfort	71.11%	32	26.67%	12	2.22%	1	0.00%	0	45
2	Fatigue	95.56%	43	4.44%	2	0.00%	0	0.00%	0	45
3	Headache	84.44%	38	13.33%	6	2.22%	1	0.00%	0	45

4	Eye Strain	73.33%	33	26.67%	12	0.00%	0	0.00%	0	45
5	Difficulty Focusing	86.67%	39	11.11%	5	2.22%	1	0.00%	0	45
6	Salivation Increase	100.00%	45	0.00%	0	0.00%	0	0.00%	0	45
7	Sweating	95.56%	43	4.44%	2	0.00%	0	0.00%	0	45
8	Nausea	93.33%	42	4.44%	2	2.22%	1	0.00%	0	45
9	Difficulty Concentrating	91.11%	41	8.89%	4	0.00%	0	0.00%	0	45
10	Blurred Vision	93.33%	42	6.67%	3	0.00%	0	0.00%	0	45
11	Dizziness with eyes open	86.67%	39	11.11%	5	2.22%	1	0.00%	0	45
12	Dizziness with eyes closed	100.00%	45	0.00%	0	0.00%	0	0.00%	0	45
13	Vertigo	93.33%	42	6.67%	3	0.00%	0	0.00%	0	45
14	Stomach Awareness	77.78%	35	15.56%	7	4.44%	2	2.22%	1	45
15	Burping	100.00%	45	0.00%	0	0.00%	0	0.00%	0	45

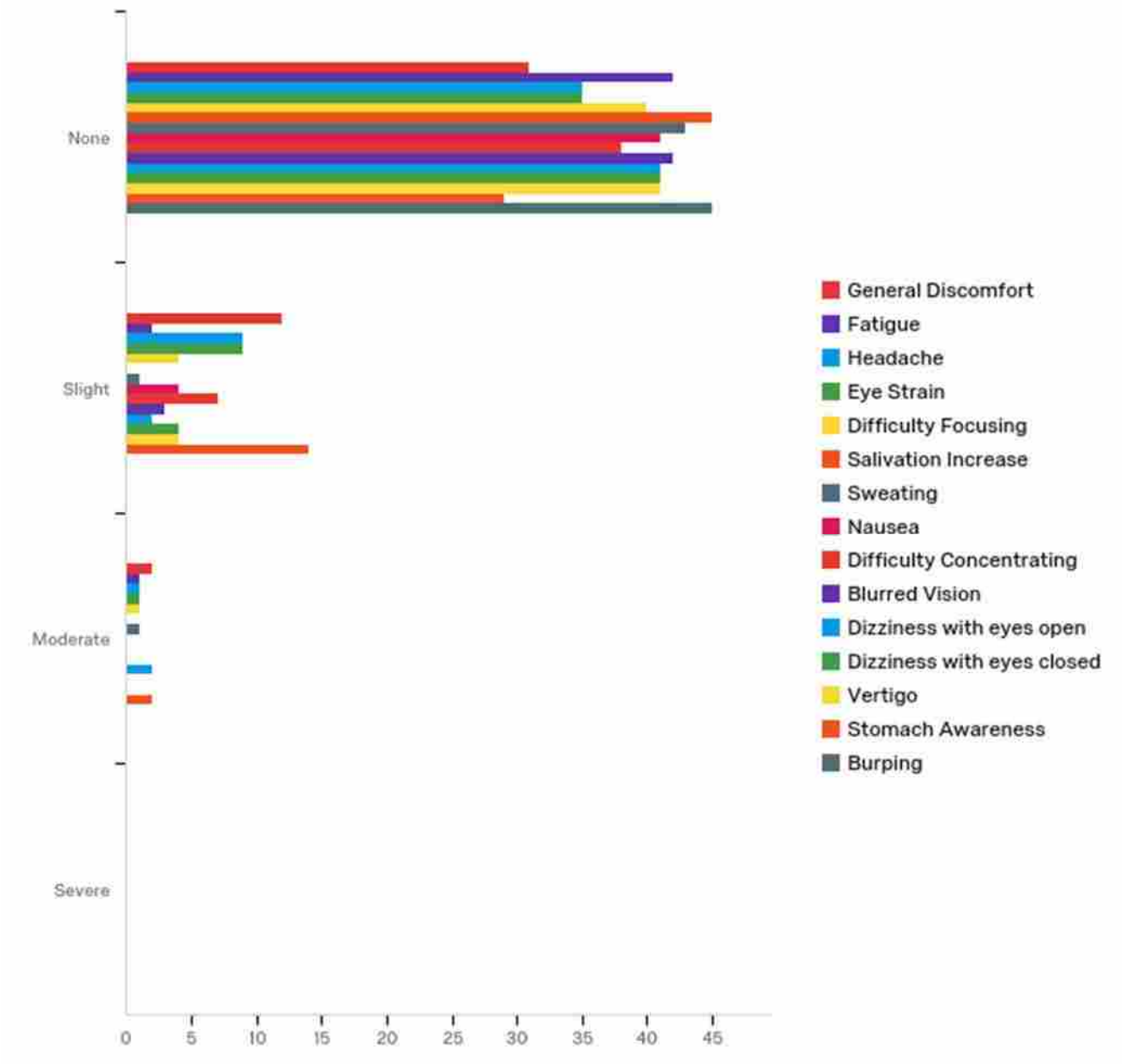
Q17 - Movement Set 2



#	Question	None		Slight		Moderate		Severe		Total
1	General Discomfort	64.44%	29	31.11%	14	4.44%	2	0.00%	0	45
2	Fatigue	97.78%	44	2.22%	1	0.00%	0	0.00%	0	45
3	Headache	77.78%	35	22.22%	10	0.00%	0	0.00%	0	45

4	Eye Strain	68.89%	31	31.11%	14	0.00%	0	0.00%	0	45
5	Difficulty Focusing	93.33%	42	6.67%	3	0.00%	0	0.00%	0	45
6	Salivation Increase	91.11%	41	8.89%	4	0.00%	0	0.00%	0	45
7	Sweating	97.78%	44	2.22%	1	0.00%	0	0.00%	0	45
8	Nausea	86.67%	39	11.11%	5	2.22%	1	0.00%	0	45
9	Difficulty Concentrating	91.11%	41	8.89%	4	0.00%	0	0.00%	0	45
10	Blurred Vision	93.33%	42	6.67%	3	0.00%	0	0.00%	0	45
11	Dizziness with eyes open	88.89%	40	11.11%	5	0.00%	0	0.00%	0	45
12	Dizziness with eyes closed	97.78%	44	2.22%	1	0.00%	0	0.00%	0	45
13	Vertigo	84.44%	38	15.56%	7	0.00%	0	0.00%	0	45
14	Stomach Awareness	77.78%	35	17.78%	8	4.44%	2	0.00%	0	45
15	Burping	93.33%	42	6.67%	3	0.00%	0	0.00%	0	45

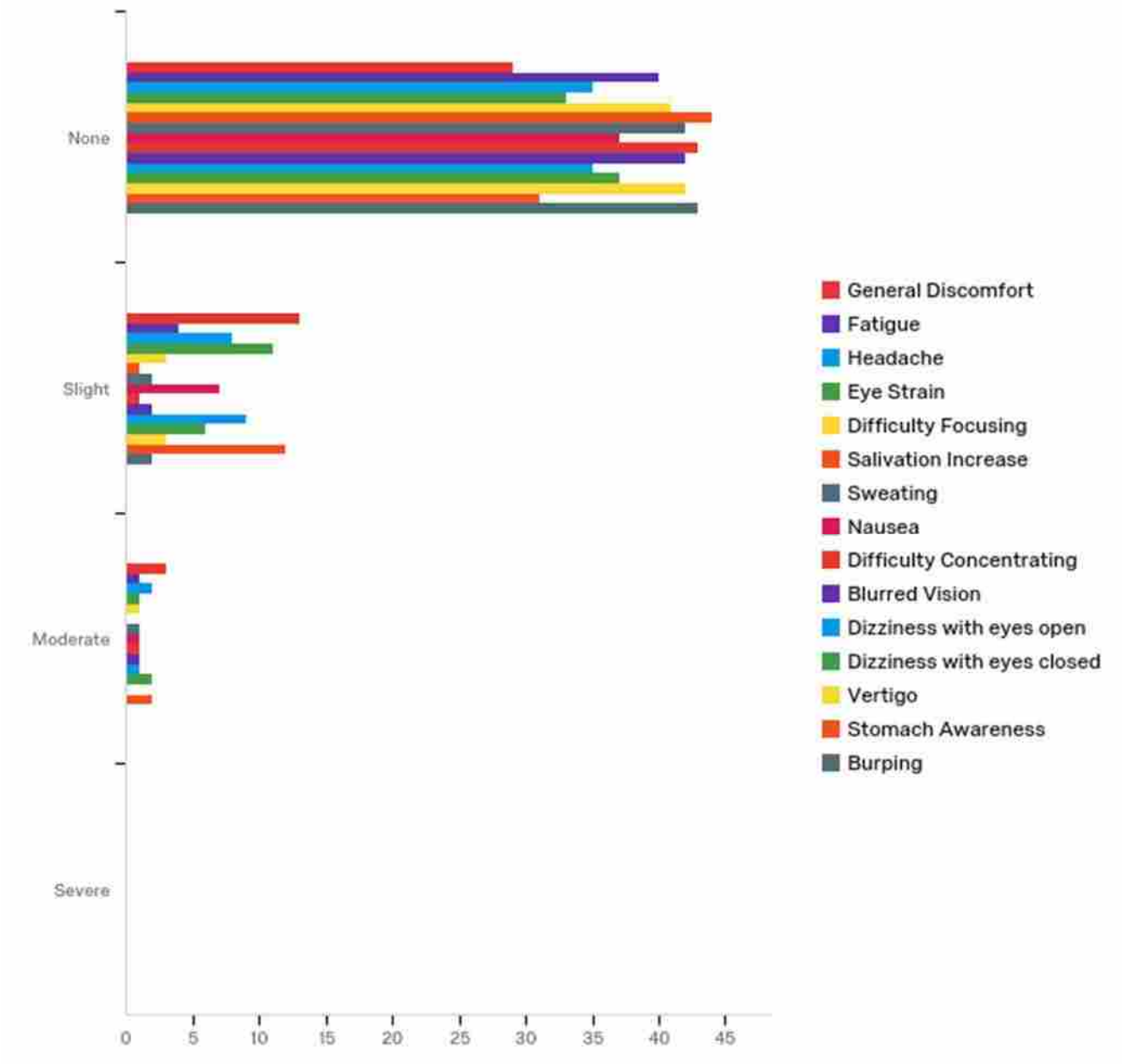
Q18 - Movement Set 3



#	Question	None		Slight		Moderate		Severe		Total
1	General Discomfort	68.89%	31	26.67%	12	4.44%	2	0.00%	0	45
2	Fatigue	93.33%	42	4.44%	2	2.22%	1	0.00%	0	45
3	Headache	77.78%	35	20.00%	9	2.22%	1	0.00%	0	45

4	Eye Strain	77.78%	35	20.00%	9	2.22%	1	0.00%	0	45
5	Difficulty Focusing	88.89%	40	8.89%	4	2.22%	1	0.00%	0	45
6	Salivation Increase	100.00%	45	0.00%	0	0.00%	0	0.00%	0	45
7	Sweating	95.56%	43	2.22%	1	2.22%	1	0.00%	0	45
8	Nausea	91.11%	41	8.89%	4	0.00%	0	0.00%	0	45
9	Difficulty Concentrating	84.44%	38	15.56%	7	0.00%	0	0.00%	0	45
10	Blurred Vision	93.33%	42	6.67%	3	0.00%	0	0.00%	0	45
11	Dizziness with eyes open	91.11%	41	4.44%	2	4.44%	2	0.00%	0	45
12	Dizziness with eyes closed	91.11%	41	8.89%	4	0.00%	0	0.00%	0	45
13	Vertigo	91.11%	41	8.89%	4	0.00%	0	0.00%	0	45
14	Stomach Awareness	64.44%	29	31.11%	14	4.44%	2	0.00%	0	45
15	Burping	100.00%	45	0.00%	0	0.00%	0	0.00%	0	45

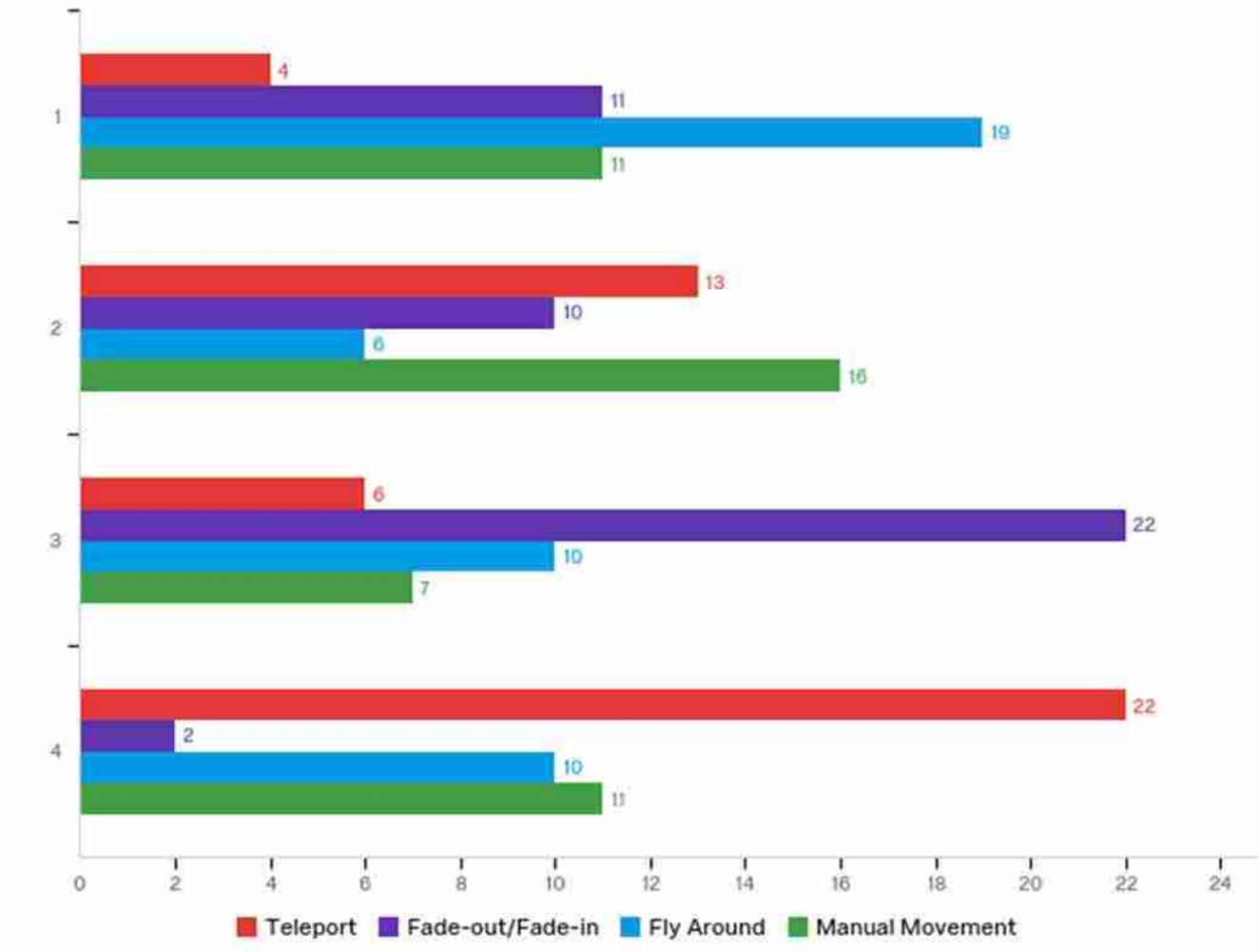
Q19 - Movement Set 4



#	Question	None		Slight		Moderate		Severe		Total
1	General Discomfort	64.44%	29	28.89%	13	6.67%	3	0.00%	0	45
2	Fatigue	88.89%	40	8.89%	4	2.22%	1	0.00%	0	45
3	Headache	77.78%	35	17.78%	8	4.44%	2	0.00%	0	45

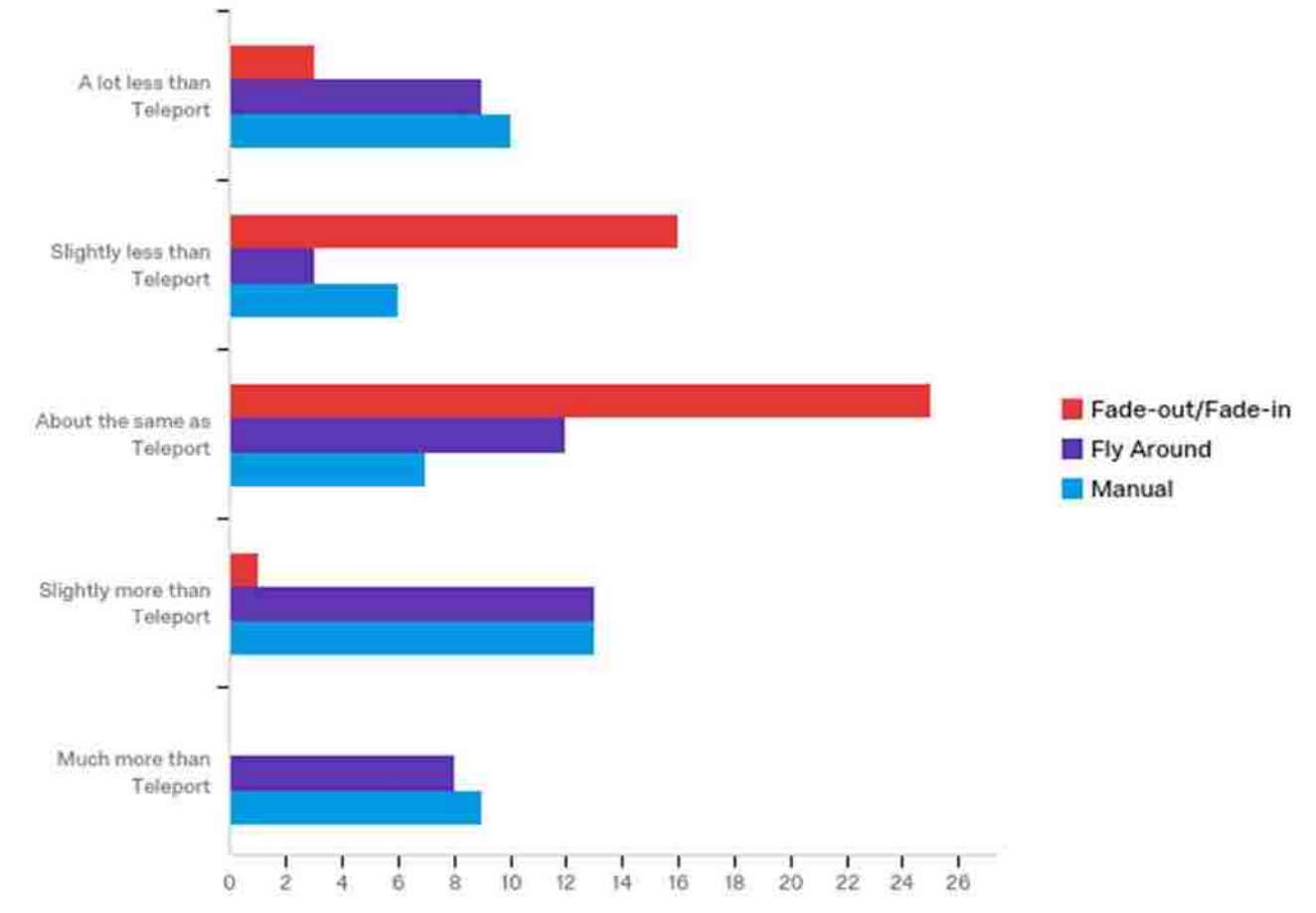
4	Eye Strain	73.33%	33	24.44%	11	2.22%	1	0.00%	0	45
5	Difficulty Focusing	91.11%	41	6.67%	3	2.22%	1	0.00%	0	45
6	Salivation Increase	97.78%	44	2.22%	1	0.00%	0	0.00%	0	45
7	Sweating	93.33%	42	4.44%	2	2.22%	1	0.00%	0	45
8	Nausea	82.22%	37	15.56%	7	2.22%	1	0.00%	0	45
9	Difficulty Concentrating	95.56%	43	2.22%	1	2.22%	1	0.00%	0	45
10	Blurred Vision	93.33%	42	4.44%	2	2.22%	1	0.00%	0	45
11	Dizziness with eyes open	77.78%	35	20.00%	9	2.22%	1	0.00%	0	45
12	Dizziness with eyes closed	82.22%	37	13.33%	6	4.44%	2	0.00%	0	45
13	Vertigo	93.33%	42	6.67%	3	0.00%	0	0.00%	0	45
14	Stomach Awareness	68.89%	31	26.67%	12	4.44%	2	0.00%	0	45
15	Burping	95.56%	43	4.44%	2	0.00%	0	0.00%	0	45

Q2 - Please Rank you movement style preference for the task performed, with 1 being your most preferred style and 4 being your least preferred style.



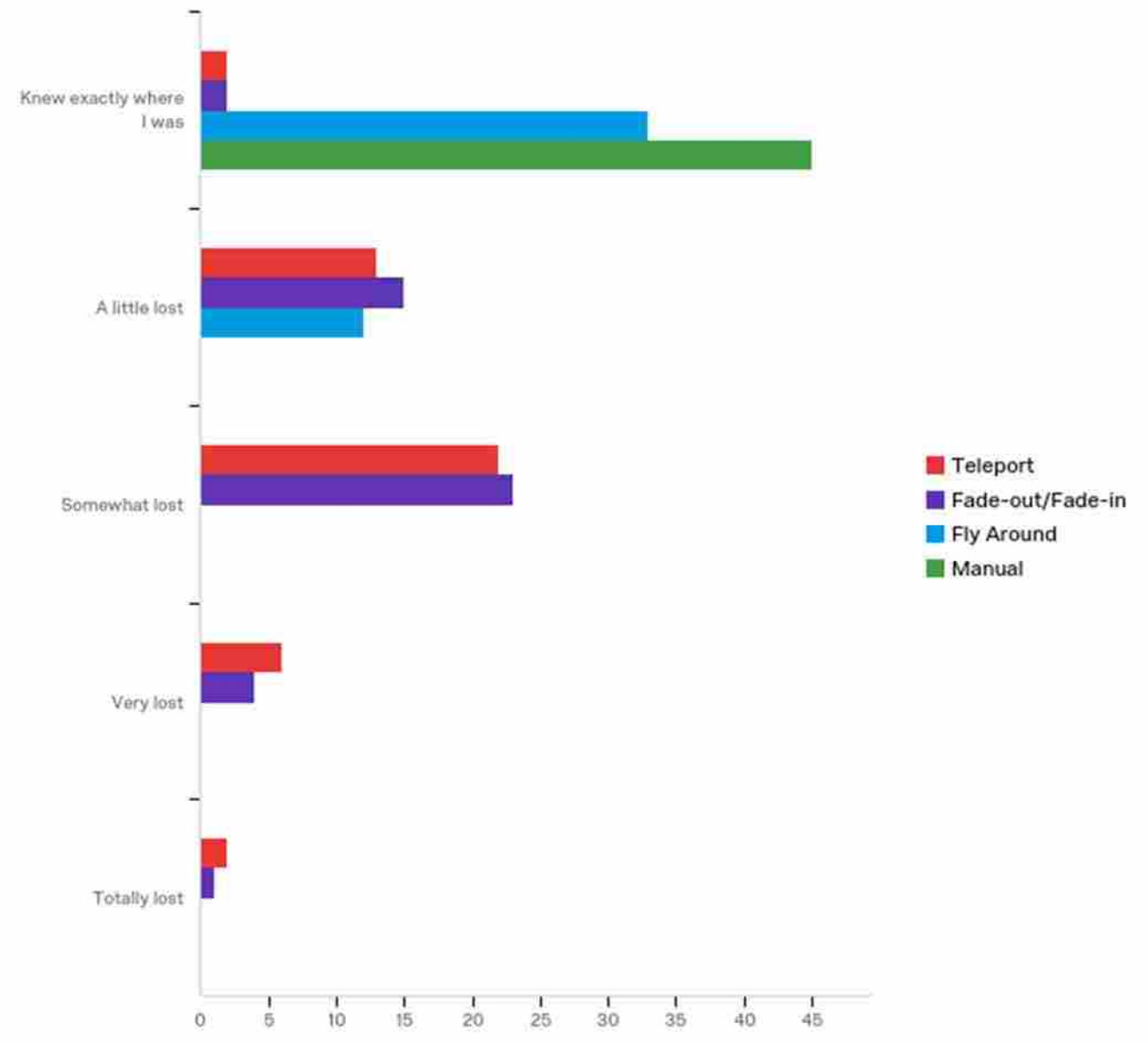
#	Question	1		2		3		4		Total
1	Teleport	8.89%	4	28.89%	13	13.33%	6	48.89%	22	45
2	Fade-out/Fade-in	24.44%	11	22.22%	10	48.89%	22	4.44%	2	45
3	Fly Around	42.22%	19	13.33%	6	22.22%	10	22.22%	10	45
4	Manual Movement	24.44%	11	35.56%	16	15.56%	7	24.44%	11	45

Q3 - Compared to "Teleport", how much physical discomfort (cybersickness) did you experience with the following movement styles?



#	Question	A lot less than Teleport		Slightly less than Teleport		About the same as Teleport		Slightly more than Teleport		Much more than Teleport		Total
1	Fade-out/Fade-in	6.67%	3	35.56%	16	55.56%	25	2.22%	1	0.00%	0	45
2	Fly Around	20.00%	9	6.67%	3	26.67%	12	28.89%	13	17.78%	8	45
3	Manual	22.22%	10	13.33%	6	15.56%	7	28.89%	13	20.00%	9	45

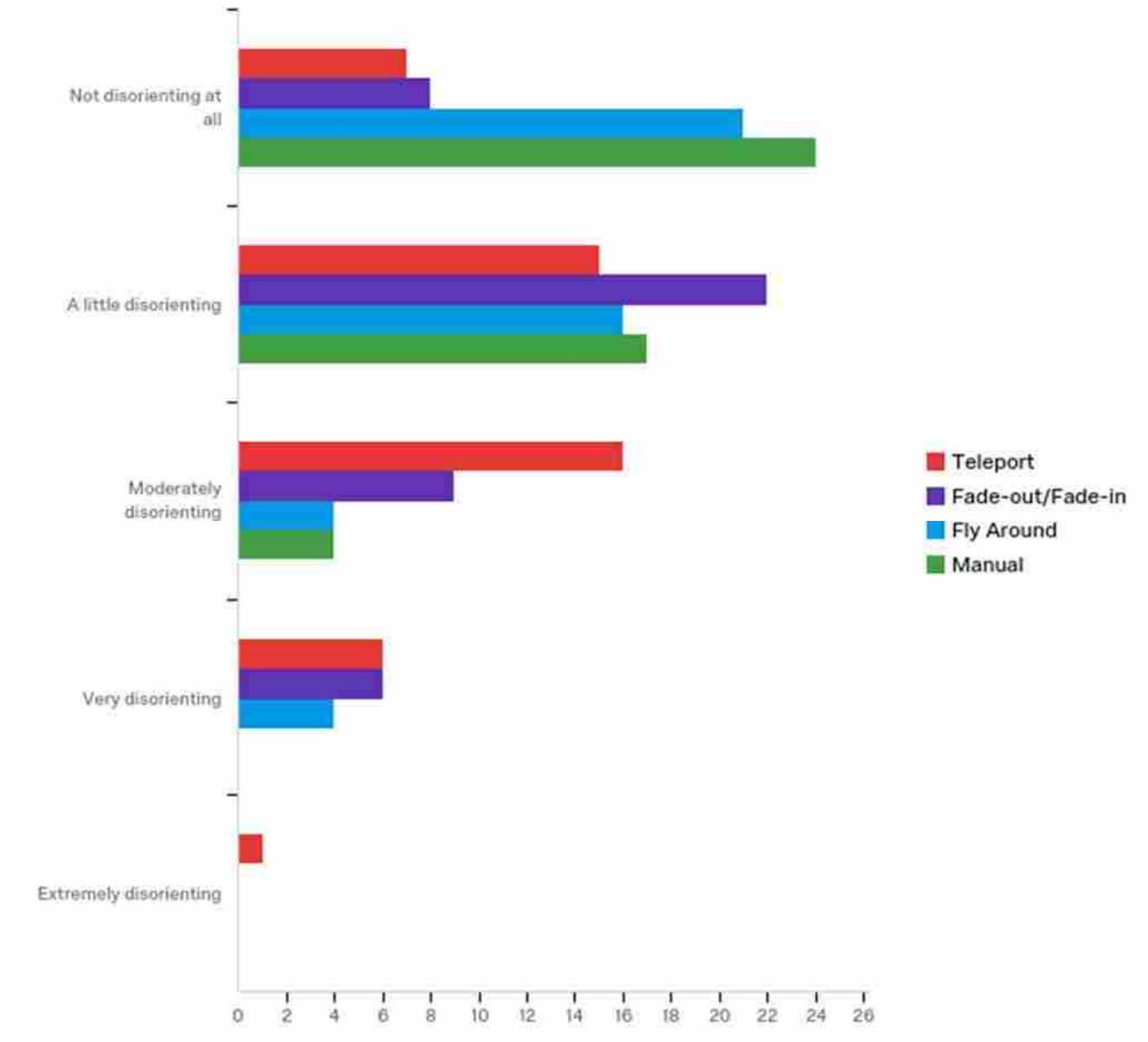
Q5 - In general, how lost did you feel after immediately following a movement?



#	Question	Knew exactly where I was		A little lost		Somewhat lost		Very lost		Totally lost		Total
1	Teleport	4.44%	2	28.89%	13	48.89%	22	13.33%	6	4.44%	2	45
2	Fade-out/Fade-in	4.44%	2	33.33%	15	51.11%	23	8.89%	4	2.22%	1	45

3	Fly Around	73.33%	33	26.67%	12	0.00%	0	0.00%	0	0.00%	0	45
4	Manual	100.00%	45	0.00%	0	0.00%	0	0.00%	0	0.00%	0	45

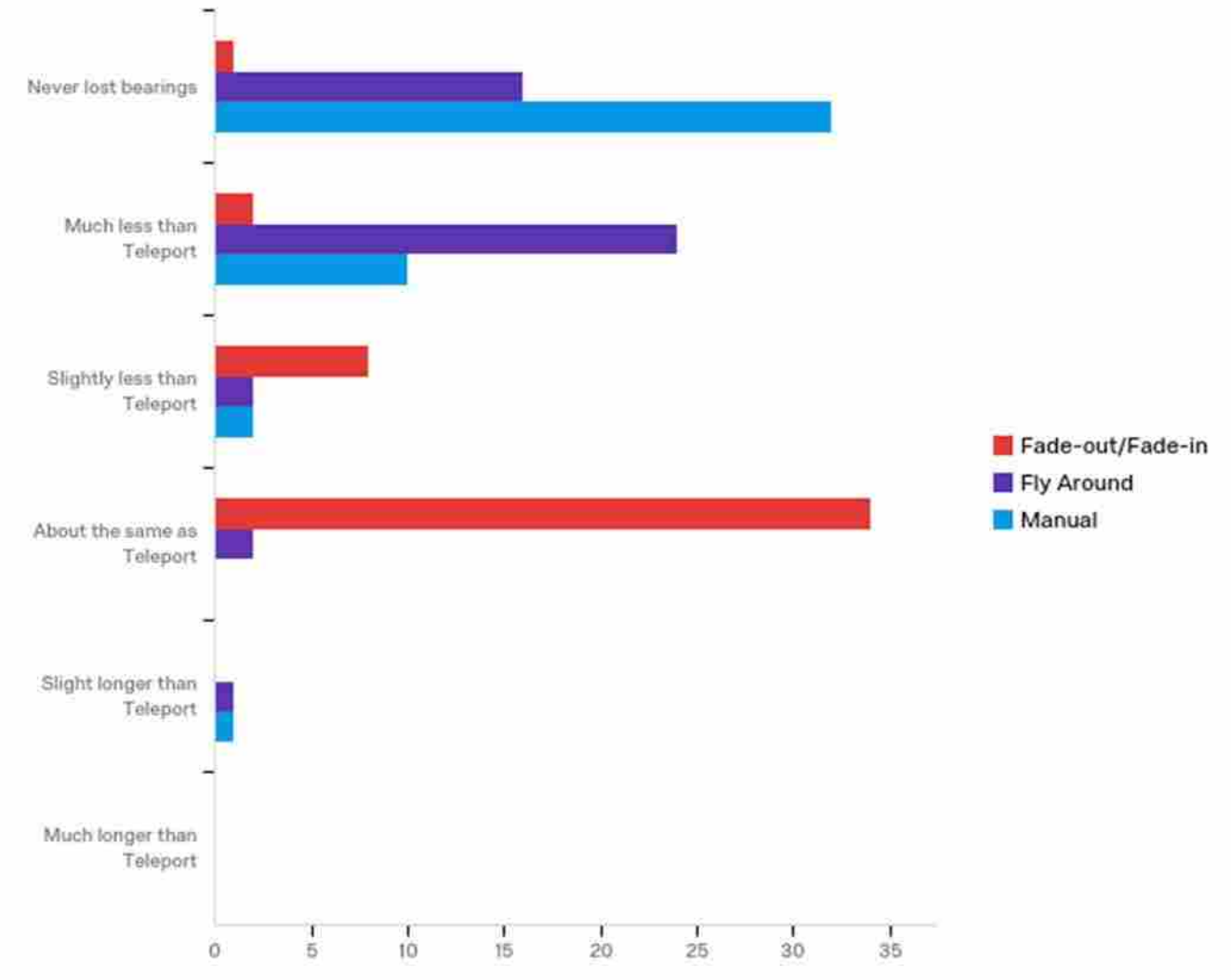
Q7 - During the actual movement, how disorienting were the following movement styles?



#	Question	Not disorienting at all		A little disorienting		Moderately disorienting		Very disorienting		Extremely disorienting		Total
1	Teleport	15.56%	7	33.33%	15	35.56%	16	13.33%	6	2.22%	1	45
2	Fade-	17.78%	8	48.89%	2	20.00%	9	13.33%	6	0.00%	0	45

	out/Fade-in				2							
3	Fly Around	46.67%	21	35.56%	16	8.89%	4	8.89%	4	0.00%	0	45
4	Manual	53.33%	24	37.78%	17	8.89%	4	0.00%	0	0.00%	0	45

Q12 - Compared to "Teleportation", how long did it take for you to regain your bearings after a movement of the following styles?



#	Question	Never lost bearings		Much less than Teleport		Slightly less than Teleport		About the same as Teleport		Slight longer than Teleport		Much longer than Teleport		Total
1	Fade-out/Fade-in	2.22%	1	4.44%	2	17.78%	8	75.56%	34	0.00%	0	0.00%	0	45
2	Fly Around	35.56%	16	53.33%	24	4.44%	2	4.44%	2	2.22%	1	0.00%	0	45

3	Manual	71.11%	32	22.22%	10	4.44%	2	0.00%	0	2.22%	1	0.00%	0	45

Q8 - Why is "[QID2-ChoiceGroup-ChoiceWithLowestValue]" your preferred movement style for the task?

Why is "[QID2-ChoiceGroup-ChoiceWithLowestValue]" your preferred movement s...
It gets me directly to the location while "zooming" out, so I can view the car from a distance and help me keep my bearings as it moved in on the part to be discussed.
It was a quick transition and I always had a decent view to what i was looking at. It was very similar to Transport.
Comfortable transition - able to get my bearings after arriving at the new location and it was easy on the eyes.
the pause in sensation created the feeling of a cut in a movie, with out the spinning swirling of flying or manual navigation.
I can see where I'm going and orient myself with what I can see of the car.
It was easier to move around than manual, but helped me orient myself in relation to other parts of the car.
Is was a gentle transition
Because it was automatic AND allowed me to see how my view was changing, this was preferred. I will say there were a few transitions that seemed confusing. For example, it flew me out and then rotated the car when it seemed like a more direct line to the new orientation would have been preferred.
I can know exactly how to position myself and exactly where I am at.
I preferred to find the angle myself instead of the default angle given. Although the controls were a slightly difficult to master, I still preferred it to the fade in/fade out because I could view from the angle I preferred.
It allowed me to regain my bearings if I forgot exactly where I was.
I prefer manual movement because I always felt in control of where I was.
I knew exactly where I was all the time and could change my view to best describe the location of the blinking object
It gave an overview of the part and you could easily tell where you came from and where you were going.
Least amount of cyber sickness. was rarely very difficult for me to regain bearings
I like how it maintains my orientation in a way that is predictable. I feel like it is the best method for beginners to VR. Once a person gets more experience, manual would offer much more flexibility though and the subject would be comfortable with it.
Not as jerky as the other ones. Less sick
I could control the situation and movement. I liked knowing where I was as I moved around.
It was similar to teleport, because i didn't have to do anything and there was no swooshing and flying and spinning. But I thought it was less abrupt/ jarring.

It's easier on the eyes than teleport. And the controls on manual were sometimes a little hard to use.

It brought me to the location of the part without making me use the controls I was unused to. This allowed me to see where I was, how I got there, and gave me good bearings.

Fly around is my preferred movement style because I know where I am coming from, and it has much more control over the movement then I do.

The movement of the screen when flying, whether automatic or manually, was disorienting at times.

I like being in control and being able to explore.

More easier to orientate myself, and feel more comfortable.

When I move manually, I know where I am going and I can compare with where I am going to where I currently am.

It took considerably less time to reorient after movement because you could seed where you were going, but didn't have the learning curve problems Manual did

I can see how I'm arriving at a certain area on the car without having to take the time to get there manually

If it had a slightly longer fade time, it is far more comfortable, less jarring, and provides a significant increase in comfort, and decrease in nauseousness.

I was able to easily maintain my bearings while not having to search around for the part myself.

You don't have to figure out the controls to get where you want to go. It feels easier on the eyes when there is a transition. Not as abrupt.

It allows me to see where the view location is but it's also fast and I don't have to spend as long finding the view as with "manual". I also don't have to worry about the desired view being hidden somewhere where I have trouble finding it (for the manual movement style) or somewhere too confusing to get bearings quickly (for the teleport/fade movement styles).

I was taken directly to the area in question and didn't have to spend time searching for it, but I also kept my bearings about where on the car I was and didn't have the object just placed in front of me arbitrarily

Fade in/fade out was nearly identical to me. Fly around was was disorienting (sensory-wise) when you went through objects. Manual movement controls were a little unnatural to me.

It was quicker and easier than manual, and I knew where I was at all times, unlike fade/teleport

I was able to control where I was going and knew exactly where I was. I could also go at my own speed.

Movement styles where I moved in the program made me feel sick and teleportation was a bit more disorienting then this.

I felt a little less disorientated using fade-out/ fade in compared to teleport and I felt little to no cyber sickness while with the manual and fly around movements the cyber sickness was distracting.

I control the movement and it is easy to slow down if needed and I don't lose bearings

I liked knowing where I was going. It took a lot less effort to answer the questions because I was oriented.

Because I always never exactly where I was, because I could see where I was coming from. And I never had a vertigo sensation like in manual.

It's faster than manual, but still allows me to see where I am moving to from where I was.

It hardly made me cyber sick until after I took the googles off my head. However the slight sickness I felt after might have been from a combination of all four of the movements.

I felt it was easier to remain bearings while it was also easy to get turned upside down in the other ones

For the purpose of locating a part, or matching someone else's view, it was quicker than manual, and I didn't lose my bearings during it.

Q9 - Why is "[QID2-ChoiceGroup-ChoiceWithHighestValue]" your least preferred movement style for the task?

Why is "[QID2-ChoiceGroup-ChoiceWithHighestValue]" your least preferred mov...
My ability to move myself to the desired location was cumbersome and sometimes disorienting. Also, it took much longer because I sometimes couldn't see the cone or part from my current vantage point.
Because of controls, I sometimes would push the wrong way and had to take more time to readjust, and my view angle was often not a preferred orientation for me to look / move around. It was also just slower.
Movement is the easiest to keep track of where you are I thought, however, getting the controls to work the way I wanted was sometimes disorienting when I pressed the wrong direction. If the car spun a different direction than I meant for it to (by my pressing the wrong direction) I felt slightly disoriented.
the abrupt change in speed, and the relatively intense rotation as the car moved around me, was rather uncomfortable
Poof, I'm there and I have to look around and reorient myself.
It required me to think a lot about where I was a look around. It took more effort.
My eyes told me different things than my inner ear
I felt it was so jarring to sudden end up in a location and orientation and required me to spend some time to get my bearings.
Most difficult for orientation and view.
It was far too fast paced and would make me motion sick. Often times the pink cone would be going right through my eye line which would cause discomfort.
Much too sudden.
I disliked it because of the time it took me to get oriented after teleporting.
It was just weird to fade in and out. Because it was like teleport, just slower
It took the longest and you had to find the position yourself.
highest cyber sickness, was also so fast I often felt more disoriented by the motion
I don't like how it instantly puts me in a certain orientation that I am not ready for and this causes vertigo and disorientation.
I got disoriented really easy. Felt a little sick afterwards
The speed of the movement at times was disorienting and my eyes had a harder time focusing.
it was kind of hard to keep track of the controls because the bearings would change when I looked around. And because it was hard to control, there was a lot of turning and spinning and it felt like falling and made me nauseous.
Because it made me feel somewhat dizzy.

If it took me somewhere immediately without any frame of reference to know exactly where I was. If it was with an object I was less familiar with than a car it would have been much harder.

Because you just appear in a section and you have no clue where you are, or which way will be up.

This method was the most disorienting for me, especially when it would fly through other components of the car.

It was very sudden/abrupt and sometimes disorienting.

Feel sick when moving around.

It is very disorienting and at times I have no idea where I ended up. I needed to look around a lot more to gain my bearings.

Both teleport and fade in/fade out required some time to figure out where I was, teleport was slightly more jarring due to the sudden transition, but not by much

It took forever to realize where I was, and was a little bit jarring visually

It was incredibly jarring and nauseating, much akin to carsickness. The flying itself was mediocre, but the start and end were particularly bad.

It just look the longest to re-orient myself, because I magically appeared in a random location.

It goes really fast and makes it hard to follow it. Makes you a little sick when it flies around so much.

Although it's fast, it gives me no time to find my bearings and it doesn't show me where I am very clearly. It's also worse than fade because fade gives me a bit of time to prepare for a completely different view.

Sometimes you were dropped in a random part of the car and had to look around to get your bearings, which was pretty annoying

I would have liked to be able to reconfigure the controls. For example, I would have switched the triggers to zoom, and had a control to roll the view.

It was very disorienting and I was less sure of my location

It was the most disorienting and caused the most strain on my eyes when the screen would change so quickly.

It made me feel dizzy, especially when I accidentally used the controller incorrectly, causing myself to move in a different direction than anticipated.

I felt a little trapped in the movement and when I moved with the controller the resulting movements were often slightly different than what I had expected which contributed to the cyber sickness.

Very disorienting, dizziness during the movement, most shocking to the body, felt like motion sickness.

I was disoriented by jumping to a new area. Even though I didn't have as much cyber sickness during it, it took a lot more concentration and time to answer the questions.

Because sometimes I could not tell where I was.

I have to piece together my location by looking around. I feel reorienting myself made things take a bit longer

Manual movement had the most cyber sickness involved and I had a hard time getting the car to do what I wanted it to do.

I enjoyed it although i couldn't tell where the stick was or where it was pointing. I also think that it was hard to get the car to be right side up after being upside down.

Most often lost my bearings in this one.

Q10 - What did you like or dislike (if anything) about the other movement styles?

What did you like or dislike (if anything) about the other movement styles?
I preferred teleport to the fade in/fade out.
Flying was nice to see the whole model and then zoom in to where you were looking, it would be nice to see where others were viewing if they were looking at different parts of the model.
I liked the flyaround view, I thought it would be most helpful if it were a little slower.
I liked moving, because I was able to understand the car better, especially as I saw how the parts of the car were put into place. I disliked maunal because while navigating around the car, I sometimes spun myself further then I intended to, which made cyber sickness worse
Manual needs roll control. Fade in/out should stay black for at least half a second longer. Flying should be slightly slower.
Manual and fly around were the best because I could see the whole car and the cone pointing me to where to focus. The other two styles went directly to the point of focus, and I wasn't able to immediately understand where I was.
Manuals roll control didn't have all axis
With manual I would sometimes end up in a weird orientation and couldn't figure out an easy way to change my location. Manual also took more time to location where the flashing part was. That being said, I liked the ability to control my orientation rather than being limited to just turning or moving my head as in other styles. I did prefer fade in over teleport because the fading made things less abrupt.
For teleport and fade in fade out there was little to no fatigue or motion sickness where as with the others, there was slight.
The only thing I didn't like about the other type of my movement was the angles that were provided. Often times, it would be too close to the part or would just have a difficult angle to view the item.
Fly Around found exactly where I needed to be for me.
I didn't mind the flying style because I generally knew where I was at and it was faster than manual movement.
Kind of disoriented you a little bit and you had to stick your head through walls which was kind of an odd sort of idea (fly around was fine, this was really just with the teleport and fade in/out)
They were all good but for finding a position that someone else was using the fly in fly out was the easiest to know what happened.
Fade out/in didn't seem to add any benefit to teleport and took more time. Manual movement still added to my cyber sickness
I liked the manual because it gave me much more freedom to move around. However, it was hard to keep myself in a level horizontal setting, which I did not like. For the fade in and out, it was less abrupt than teleport, which I liked, but it too placed me in unexpected orientations, and I didn't really notice a difference from teleport.
Manual gives a lot of control, but makes me a little sicker than fade-in and out

I liked the fade in fade out because it was a little easier on the eyes.

Fly was a lot more disorienting than I thought it was going to be, and it made me feel a little bit more sick than I was expecting. It spun me around in ways I couldn't anticipate and it was quite fast. Teleport was easier than I thought it would be. I liked how I could peak my head up through a motor or a door to see around, that made it quite easy to find my bearings.

For manual style I liked being able to control things.

I liked that Manual was slower than Fly Around and it allowed me to change how close I was to something and change the angle I was looking at it from. But the hard part was sometimes getting to the object and having the car oriented the way I wanted to once I got there.

I liked being able to control my movement, but it was difficult to control. The flyover provided a much easier way to see the car the way I wanted to.

The manual method is best for always being oriented, but maybe slow the movement down a little. It responds a little too quickly.

The fly method was cool because it was seamless and the manual method was the best because you could choose the view/perspective.

Teleport make me lost some time

I like that the fly around is a lot faster than manual, but manual gives me a sense of freedom to go where I want. Teleport and fade in/fade out were very disorienting.

The manual style was actually quite useful and could potentially be the most effective if someone was given enough time to become proficient at the controls

Manual is nice because I have control over the angle at which I can approach the object

Teleportation was slightly jarring, in that it took a second to register a new view, and then a second or two to figure out where it was.

Moving around was really easy to stay oriented, but took some getting used to the movement controls themselves. Fade in and out didn't really make a huge difference from teleport, however if I didn't know the change was going to happen, I would have preferred the fade to the teleport function.

I liked manual because you could choose what you do and stop if you need to reorient yourself.

Manual movement was good for knowing where I was but it took significantly longer. Fading in and out was good because it was less sudden but it unfortunately didn't let me see where I was going because it's still so sudden.

the only problem with the manual movement was there was no easy way to correct myself to be upright if I had tilted my viewpoint while moving, this made it a little more disorienting than usual

I would have preferred a blend of teleport and fly. That is, teleport to a zoomed out position and then fly into the view. Without going through any objects.

I liked the orientation of the manual, but it was slower than flying. Fade was only slightly better than teleport, but it was still disorienting. A longer fade would have been better

Fly was nice because I stayed pretty well oriented, but it was slightly nauseating to fly around so quickly. Fade-in/Fade-out was the easiest on the eyes.

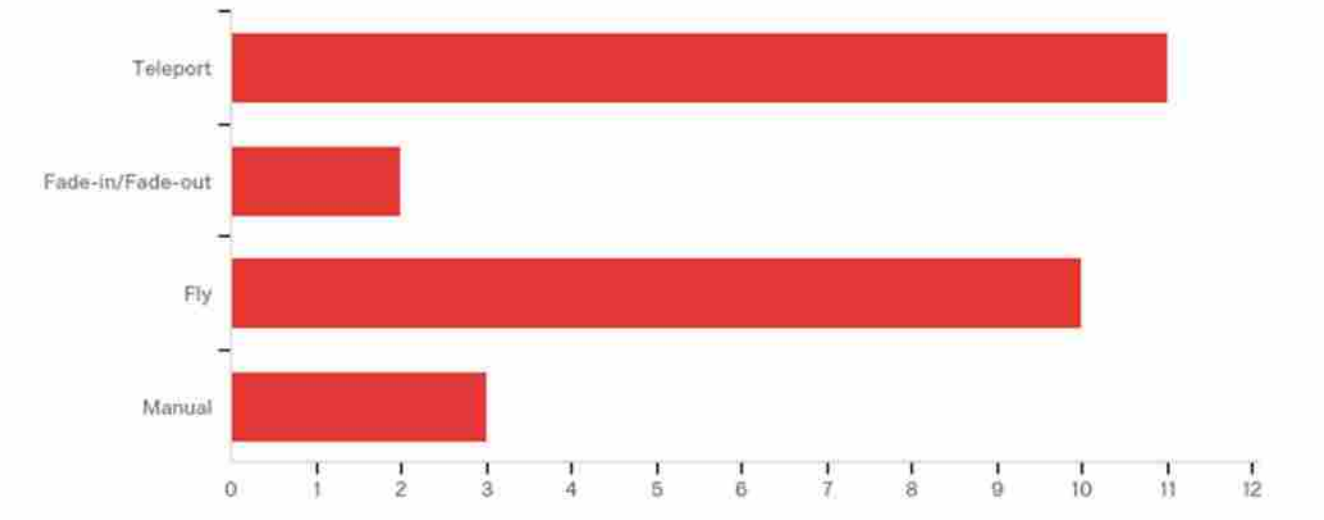
Fly around helped me know exactly where I was without as much sickness as manual, but when it gets up close to anything, I immediately felt kind of dizzy. I'm indifferent to teleportation.

Fly around helped me to keep my bearings but it did cause cyber sickness after multiple movements.

Teleport and fade in fade out were the hardest to keep bearings but haws extremely little sickness

affects, teleport is quite shocking to the mind though and is a little unpleasent.
fly around was a smooth movement for the most part which I appreciated. There was a part that was jerky which was a little disorienting because I kept getting distracted by the jerky movement.
Manual was pretty good. With practice moving around I would be able to get around easier and not feel like I was falling sideways.
In the manual movement style, the movement tilted the structure. It may be useful to have a "roll" movement option.
The thing I disliked the most was the sickness (who doesn't) the fly moment felt like it was trying to be fancy and seamed to take to long, although feels like it might be good for an initial viewing of the area but I can see it getting annoying after a wile especially when you want to quickly jump from part to part (it's cool the first or second time it is used, but after that i would want to teleport.)
sometimes it would transport me under the car and I couldn't get bearings or tell which was forward or backward.
Fade in and fade out didn't do anything different than teleport other than make it less shocking to change views.

Q11 - Would any of these movement styles be showstoppers (i.e. bad enough to cause you to avoid using the tool) if they were implemented in a VR tool?



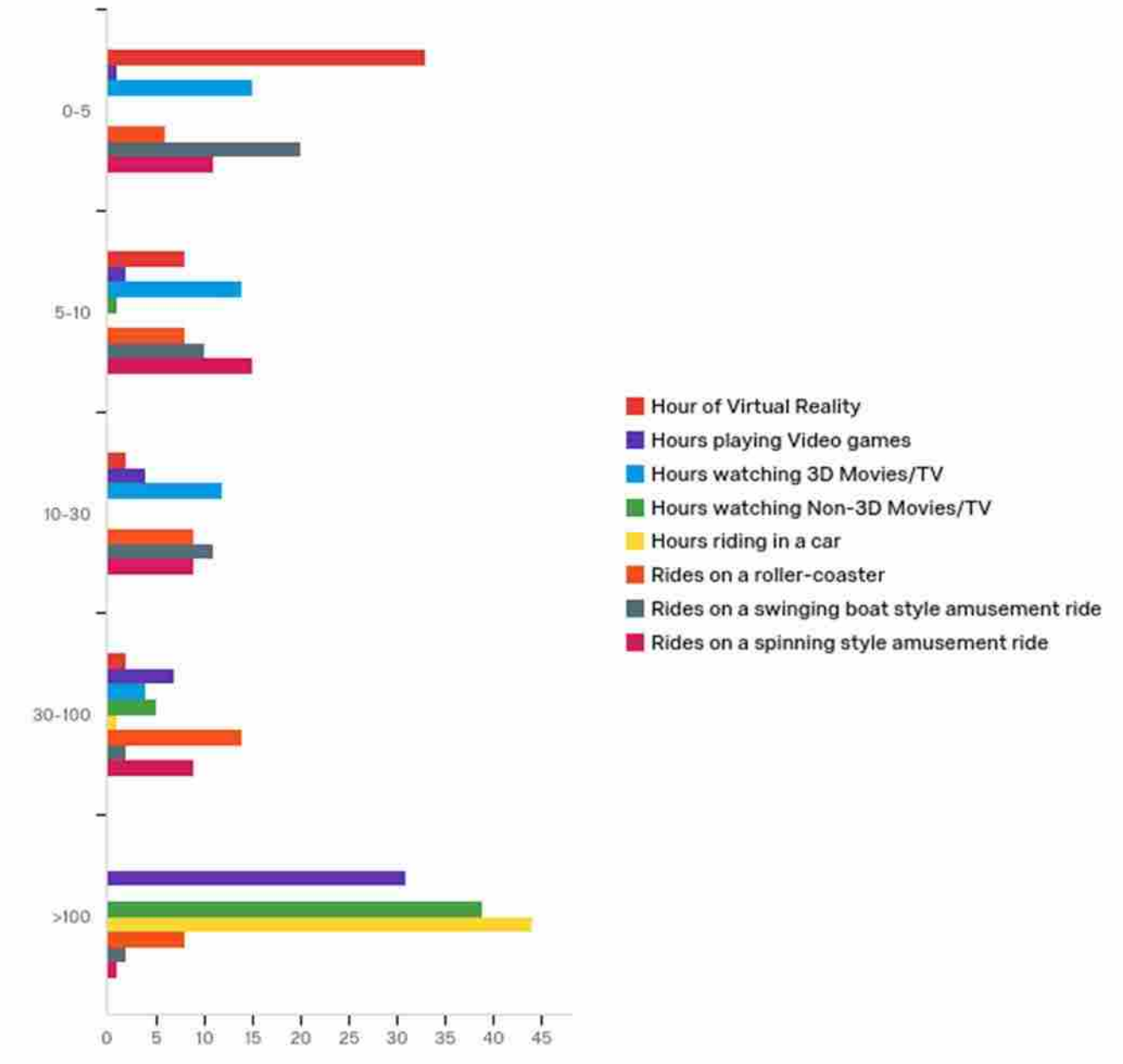
#	Answer	%	Count
1	Teleport	47.83%	11
2	Fade-in/Fade-out	8.70%	2
3	Fly	43.48%	10
4	Manual	13.04%	3
	Total	100%	23

Q11 - Given the movement styles you experienced, is there a different movement style you would believe you might prefer for the task? If so please describe it.

Given the movement styles you experienced, is there a different movement st...
Maybe getting to the exact part, and then slowly zooming out and then back in on the part. However, if the part is "obvious" from the get go, this could be annoying.
none
2 step movement, teleport away from what you are looking at and then flying in to the spot, just with car already properly orientated.
Nope
physical movement with model scaling
A standing version where I can walk around the model instead of using a controller might be interesting to explore.
The fade in/fade out, teleport might be easier to orient if you have off to the side a orientation bird's eye view map of where you are/what you are looking at.
Moving the object. A movement where you could grab the object and twist and turn it. Perhaps an assembly type movement where everything is in layers and you can select a section and it would separate itself from the object for your viewing.
No other ideas.
If there was a way you could choose the point you could rotate around that would be nice. It didn't seem to quite rotate around myself like I wanted it to.
None
The fly motion might be better if it moved in more Cartesian motions (shift x, rotate z, rotate y, shift z) rather than a smooth path
I would prefer to have control over rotation around my line of sight, so that I might be able to orient myself more easily.
fade-in and out, but with the ability to manually move around afterward for reorientation
In a group setting where a certain part needed to be identified and looked at I would prefer teleport. A quicker route to the location of the part.
I was thinking it might be nice if there was a way to incorporate some manual controls into fade-in/fade-out, so I could teleport and then maybe zoom in and zoom out to find my bearings faster.
It would be great to add a button to manual style that would automatically level out my view. I think a movement style similar to fly would be nice but instead of having my vantage point zoom around I'd like to have the car zoom away reorientate itself and zoom in.
I think that bringing the user to the object via Fly Around and then letting them have manual controls once there would be good.
I do not believe there is a better style. Just need to slow down the response from the manual movement

or slow down the fly a little.
Maybe doing the actual motions myself instead of sitting (i.e. squatting, peering into the car, etc.)
Honestly, it was a tie between flying and manual as my preferred.
The rotate control in the manual style might be easier to use if it rotated around the center point of the car rather than with the viewer as the center
Perhaps with teleport or fade, a small flag or other marker could be placed at the front, or origin, or some other point to aid in orientation.
Not that I can think of.
I think the fly one would be better if it went a little slower and moved at different angles instead of moving through the car.
Not really, although I would appreciate roll control for manual movement to make it easier. Another view that could be useful in some circumstances (especially if a desired view is on the exterior) is one that rotates around a fixed center point.
No, I think either the fly or manual movements are the best, they let me keep my bearings at all time and know exactly where I was on the car
See comments above. It would also be nice if there was a way to maintain a general vertical position. This would prevent the need for a roll control.
A movement style that faded out, gave you a view of the entire car (either top, bottom, full left side, full right side, front, or back, depending on where your next view is) so that you could see where you're about to be, and then fade again to be right in at that part. I feel like I would be more oriented, but wouldn't have to move through the car, which is the part that makes me feel dizzy.
I think that a teleport or a fade in/ fade out with the ability to simply zoom in and out from the object would not cause any cyber sickness but would allow you to orient yourself quickly if you became disoriented.
No
No
I like the fly mode. But maybe being able to zoom in or out at any moment would be helpful, especially when you get to parts inside the car.
N/A
I wouldn't mind a click and zoom. you would use manual or something like that to get you close to the object and then you could lock on to the desired object and it would use that as your folcol point. Also like illustrator you could have a box representing the object that could move the object from a far to get a good overview of the object.
Not a new style but rather a return to home button on the manual.
Combination of manual and fly, where I can just pick a view (top-down, left, etc, like regular cad) and be transported to that view. Might not be preferable to just sharing a view, but nice for looking around myself.

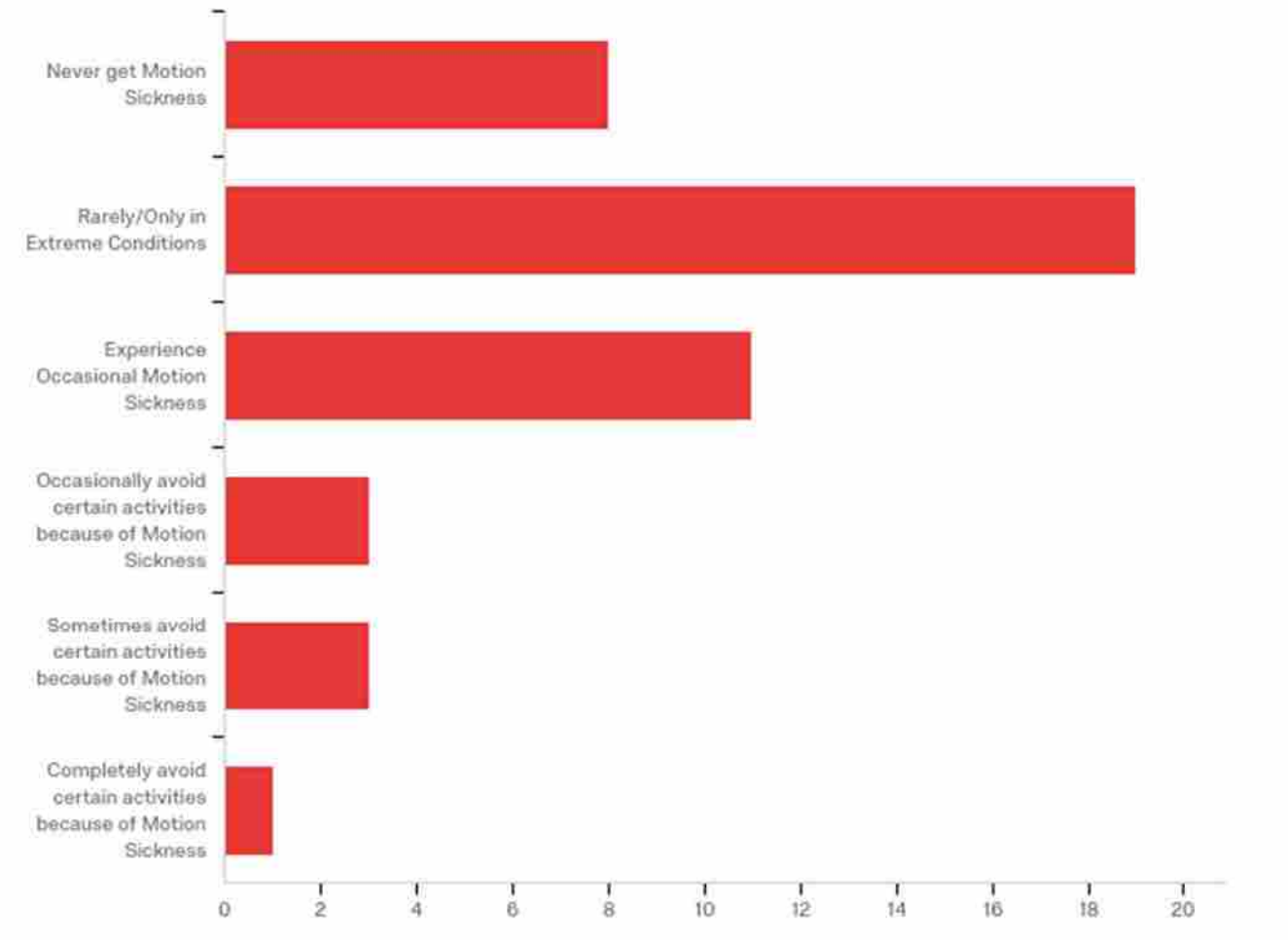
Q6 - Please list how much experience you have with the following items in your lifetime.



#	Question	0-5		5-10		10-30		30-100		>100		Total
1	Hour of Virtual Reality	73.33%	33	17.78%	8	4.44%	2	4.44%	2	0.00%	0	45
2	Hours playing Video games	2.22%	1	4.44%	2	8.89%	4	15.56%	7	68.89%	31	45

3	Hours watching 3D Movies/TV	33.33%	15	31.11%	14	26.67%	12	8.89%	4	0.00%	0	45
4	Hours watching Non-3D Movies/TV	0.00%	0	2.22%	1	0.00%	0	11.11%	5	86.67%	39	45
5	Hours riding in a car	0.00%	0	0.00%	0	0.00%	0	2.22%	1	97.78%	44	45
6	Rides on a roller-coaster	13.33%	6	17.78%	8	20.00%	9	31.11%	14	17.78%	8	45
7	Rides on a swinging boat style amusement ride	44.44%	20	22.22%	10	24.44%	11	4.44%	2	4.44%	2	45
8	Rides on a spinning style amusement ride	24.44%	11	33.33%	15	20.00%	9	20.00%	9	2.22%	1	45

Q20 - In general, how prone to Motion Sickness would you rate yourself?



#	Question	1		2		3		4		Total
1	Teleport	8.89%	4	28.89%	13	13.33%	6	48.89%	22	45
2	Fade-out/Fade-in	24.44%	11	22.22%	10	48.89%	22	4.44%	2	45
3	Fly Around	42.22%	19	13.33%	6	22.22%	10	22.22%	10	45
4	Manual Movement	24.44%	11	35.56%	16	15.56%	7	24.44%	11	45

Q22 - Do you have any other feedback you would like to share?

Do you have any other feedback you would like to share?
none
Keep up the good work!
Great experience. Well Done.
The sensitivity of the manual movement was a bit too high. I also would have liked to roll the car while I was viewing it to get a different angle and better control of my movement, as well as being able to travel up and down the Y axis without looking up and down and traveling forward.
Fly Around would be better if the path of flight was smoother.
No
NA
seemed to me like the worst motions (fly, manual) were last, as far as cyber sickness goes. Might have felt different if they were first.
Awesome survey!
got somewhat sick overall during the whole experiment. I would probably avoid VR because any kind of motion sickness even slight, is a guaranteed no for me when deciding on doing something
N/A
The ability to control roll would be nice. In general the manual mode is quite clunky, if you could move by your self more smoothly like in fly mode, that would be awesome.
This was awesome, I can't wait to see the future of the VR technology!
This was a lot of fun and I think it would be really helpful to industry so I hope your research helps it get there!
I would say the manual control would be my preferred method If i was told where to look on the car, trying to find it on my own was time consuming and sometimes difficult.
None.
No
Slow down the flying and it may become less sickness causing.
I felt slight vertigo in manual mode because I got turned sideways, and felt like I was falling sideways. If there was a way to keep my view upright, it might have helped that.
yes, I don't know if this is possible but having the ability to control the 3D plane with your hands would be awesome. also a 2D menu bar that travels with you would probably help keep you orientated you should be able to lock the menu bar or make it disappear if you wanted to.
No it was great

APPENDIX C. POST EXPERIMENTAL SURVEY FOR COLLABORATIVE VIRTUAL ENVIRONMENT COMPARATIVE STUDY

Multi-User VR Survey

Q4 Which system would you prefer to use to communicate about complex 3D data (such as the data in the task you performed) with a remote person?

- ❍ VR Environment (1)
- ❍ Skype (or similar) (3)
- ❍ Other (please explain) (4) _____

Q5 Please rate the extent to which you agree or disagree with the following statements:

	Strongly agree (1)	Agree (2)	Somewhat agree (3)	Neither agree nor disagree (4)	Somewhat disagree (5)	Disagree (6)	Strongly disagree (7)
I enjoyed using the Virtual Reality Environment. (1)	❍	❍	❍	❍	❍	❍	❍
I found the Virtual Environment distracting. (2)	❍	❍	❍	❍	❍	❍	❍
I was able to recognize communication gestures in the Virtual Environment. (3)	❍	❍	❍	❍	❍	❍	❍
I believe Virtual Reality tools could improve my engineering work. (4)	❍	❍	❍	❍	❍	❍	❍
I believe Virtual Reality tools could improve my understanding of engineering designs. (5)	❍	❍	❍	❍	❍	❍	❍
I would like to have access to Virtual Reality tools in my future workplace. (6)	❍	❍	❍	❍	❍	❍	❍

Q6 Please rate how natural or unnatural the use of communication gestures (such as pointing, waving, "talking with your hands") felt in the VR Environment.

- ❍ Very Natural (1)
- ❍ Natural (2)
- ❍ Somewhat Natural (3)
- ❍ Neither Natural nor Unnatural (4)
- ❍ Somewhat Unnatural (5)
- ❍ Unnatural (6)
- ❍ Very Unnatural (7)

Q1 Please rank the suitability of each system you used for communicating complex 3D data (such as the task you performed).

	Worthless (1)	Very Unsuitable (2)	Somewhat Unsuitable (3)	Somewhat Suitable (4)	Very Suitable (5)	Ideal (6)
Skype (1)	❍	❍	❍	❍	❍	❍
VR Environment (2)	❍	❍	❍	❍	❍	❍

Q8 Please describe what you feel makes Skype "${q://QID1/ChoiceGroup/SelectedAnswers/1}" for communicating complex 3D data.

Q9 Please describe what you feel makes the Virtual Environment "${q://QID1/ChoiceGroup/SelectedAnswers/2}" for communicating complex 3D data.

Q19 What would you like to see changed, improved, or added to the Virtual Environment if anything?

Q3 To what extent did you experience any of the following cybersickness symptoms during or after using the VR Environment?

	None (1)	Slight (2)	Moderate (3)	Severe (4)
General Discomfort (1)	❍	❍	❍	❍
Fatigue (2)	❍	❍	❍	❍
Headache (3)	❍	❍	❍	❍
Eye Strain (4)	❍	❍	❍	❍
Difficulty Focusing (5)	❍	❍	❍	❍
Salivation Increase (6)	❍	❍	❍	❍
Sweating (7)	❍	❍	❍	❍
Nausea (8)	❍	❍	❍	❍
Difficulty Concentrating (9)	❍	❍	❍	❍
Blurred Vision (10)	❍	❍	❍	❍
Dizziness with eyes open (11)	❍	❍	❍	❍
Dizziness with eyes closed (12)	❍	❍	❍	❍
Vertigo (13)	❍	❍	❍	❍
Stomach Awareness (14)	❍	❍	❍	❍
Burping (15)	❍	❍	❍	❍

Q11 Please rate how articulate your partner was when explaining the path to you.

- ❍ Extremely clear (1)
- ❍ Moderately clear (2)
- ❍ Slightly clear (3)
- ❍ Neither clear nor unclear (4)
- ❍ Slightly unclear (5)
- ❍ Moderately unclear (6)
- ❍ Extremely unclear (7)

Q16 Please rate how articulate you were when explaining the path to your partner.
- ❍ Extremely clear (1)
- ❍ Moderately clear (2)
- ❍ Slightly clear (3)
- ❍ Neither clear nor unclear (4)
- ❍ Slightly unclear (5)
- ❍ Moderately unclear (6)
- ❍ Extremely unclear (7)

Q12 Please rate how effective your partner's hand gestures in the VR Environment were in communicating the path to you.
- ❍ Extremely effective (1)
- ❍ Very effective (2)
- ❍ Moderately effective (3)
- ❍ Slightly effective (4)
- ❍ Not effective at all (5)

Q17 Please rate how effective your hand gestures in the VR Environment were in communicating the path to your partner.
- ❍ Extremely effective (1)
- ❍ Very effective (2)
- ❍ Moderately effective (3)
- ❍ Slightly effective (4)
- ❍ Not effective at all (5)

Q13 Please rate how effective your partner's mouse gestures in Skype were in communicating the path to you.
- ❍ Extremely effective (1)
- ❍ Very effective (2)
- ❍ Moderately effective (3)
- ❍ Slightly effective (4)
- ❍ Not effective at all (5)

Q18 Please rate how effective your mouse gestures in Skype were in communicating the path to your partner.
- ❍ Extremely effective (1)
- ❍ Very effective (2)
- ❍ Moderately effective (3)
- ❍ Slightly effective (4)
- ❍ Not effective at all (5)

Q11 Please list how much experience you have with the following items in your lifetime:

	0-5 (1)	5-10 (2)	10-30 (3)	30-100 (4)	>100 (5)
Hour of Virtual Reality (1)	❍	❍	❍	❍	❍
Hours playing Video games (2)	❍	❍	❍	❍	❍
Hours watching 3D Movies/TV (3)	❍	❍	❍	❍	❍
Hours watching Non-3D Movies/TV (4)	❍	❍	❍	❍	❍
Hours riding in a car (5)	❍	❍	❍	❍	❍
Rides on a roller-coaster (6)	❍	❍	❍	❍	❍
Rides on a swinging boat style amusement ride (7)	❍	❍	❍	❍	❍
Rides on a spinning style amusement ride (8)	❍	❍	❍	❍	❍

Q14 In general, how prone to Motion Sickness would you rate yourself?

- ❍ Never get Motion Sickness (1)
- ❍ Rarely/Only in Extreme Conditions (2)
- ❍ Experience Occasional Motion Sickness (3)
- ❍ Occasionally avoid certain activities because of Motion Sickness (4)
- ❍ Sometimes avoid certain activities because of Motion Sickness (5)
- ❍ Completely avoid certain activities because of Motion Sickness (6)

Q13 Do you have any other feedback you would like to share?

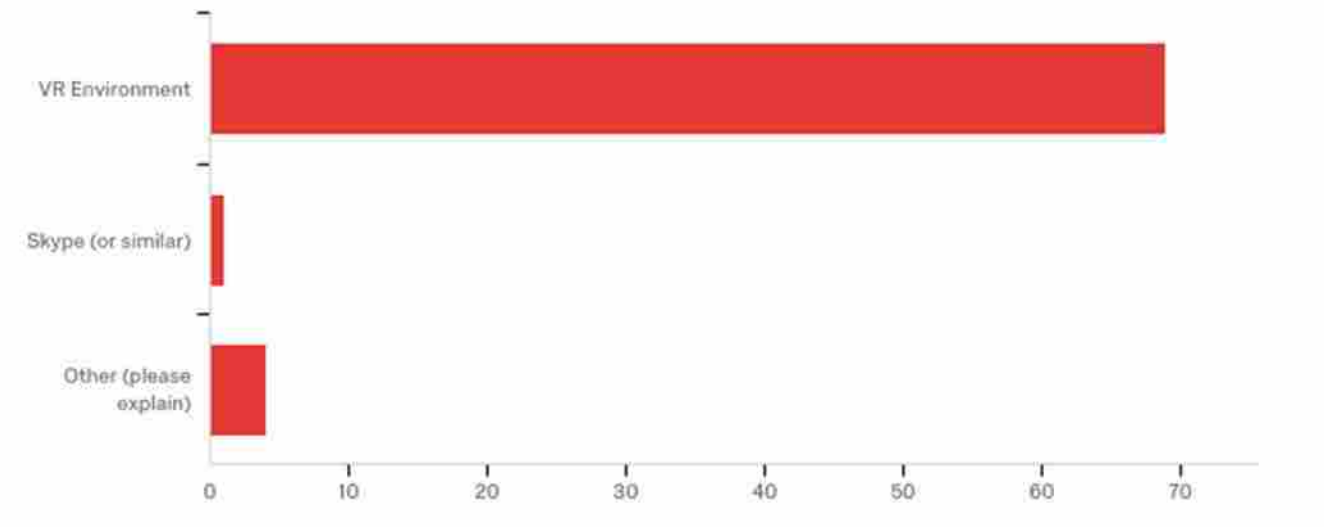
APPENDIX D. POST EXPERIMENTAL SURVEY RESULTS FOR COLLABORATIVE VIRTUAL ENVIRONMENT COMPARATIVE STUDY

Default Report

Multi-User VR Survey

June 3rd 2017, 10:48 pm MDT

Q4 - Which system would you prefer to use to communicate about complex 3D data (such as the data in the task you performed) with a remote person?



#	Answer	%	Count
1	VR Environment	93.24%	69
3	Skype (or similar)	1.35%	1
4	Other (please explain)	5.41%	4
	Total	100%	74

Other (please explain)

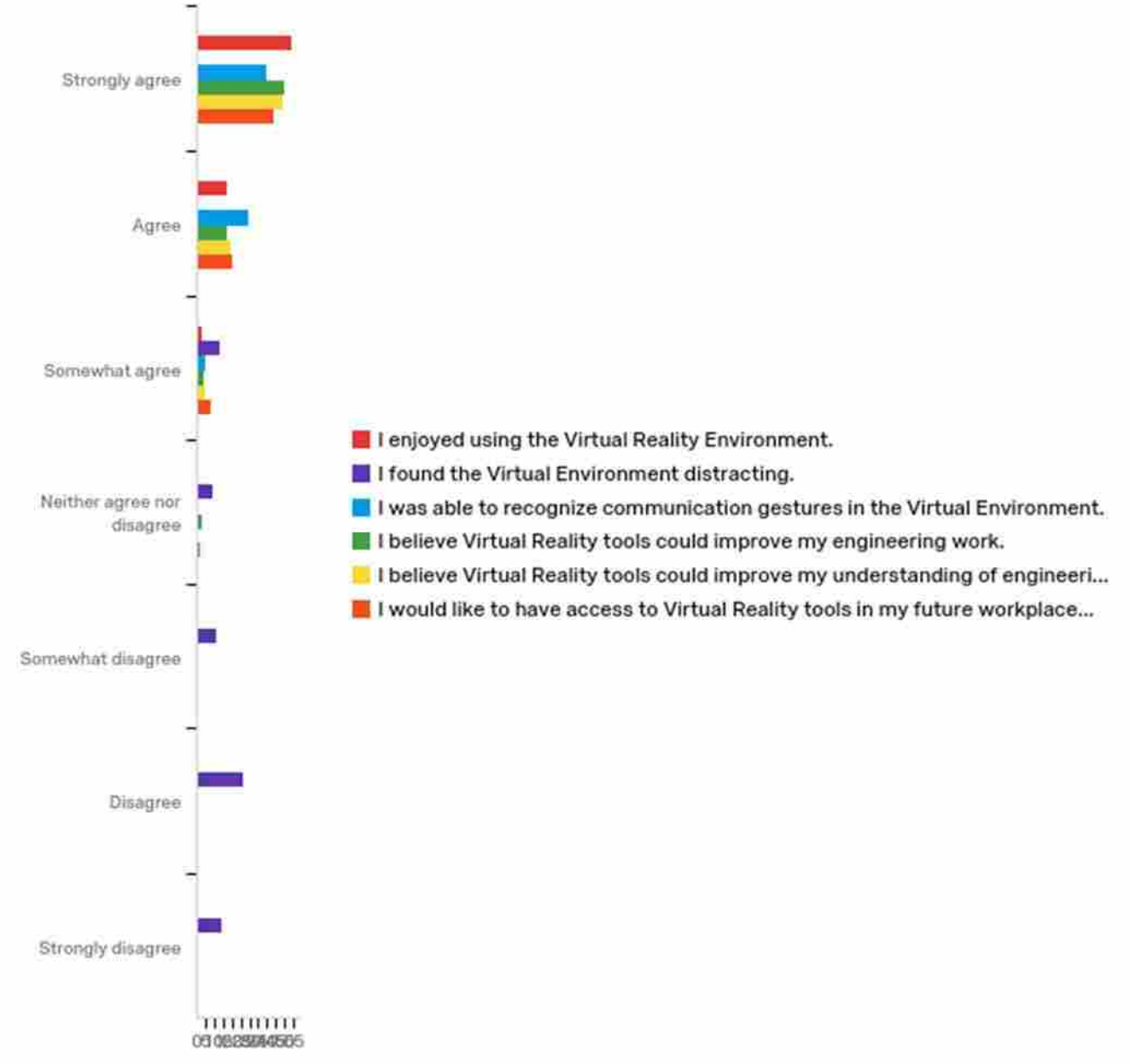
Other (please explain)
Send a picture of the model directly to the other person
Skype was great for a vertical view to get more than one angle, givnig my partner my perspective from

multiple angles. However, VR helped draw a line to show that type of visual aid.

I would prefer being in the same room with someone and using pipe cleaners or an actual 3D model I can manipulate. Play-Dough, if necessary, or toothpicks, or stuff like that.

I would like to draw it with a paper and pencil. I should have thought of this during the experiment!

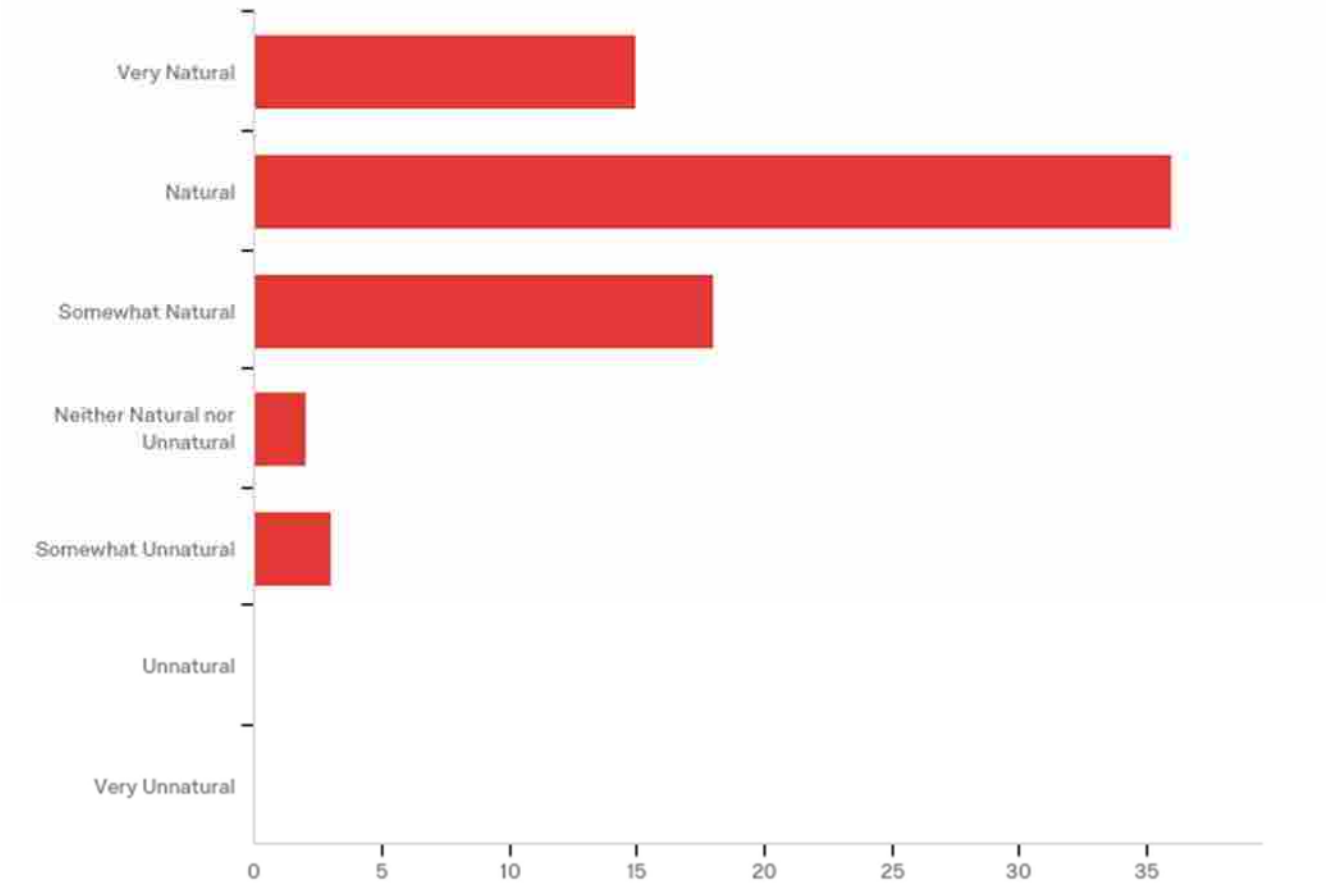
Q5 - Please rate the extent to which you agree or disagree with the following statements:



#	Question	Strongly agree		Agree		Somewhat agree		Neither agree nor disagree		Somewhat disagree		Disagree		Strongly disagree		Total
1	I enjoyed using the Virtual Reality Environment.	72.97%	54	22.97%	17	4.05%	3	0.00%	0	0.00%	0	0.00%	0	0.00%	0	74
2	I found the Virtual Environment distracting.	0.00%	0	1.35%	1	17.57%	13	12.16%	9	14.86%	11	35.14%	26	18.92%	14	74
3	I was able to recognize communication gestures in the Virtual Environment.	54.05%	40	39.19%	29	6.76%	5	0.00%	0	0.00%	0	0.00%	0	0.00%	0	74
4	I believe Virtual Reality tools could improve my engineering work.	67.57%	50	22.97%	17	5.41%	4	4.05%	3	0.00%	0	0.00%	0	0.00%	0	74
5	I believe Virtual Reality tools could improve my understanding of	66.22%	49	25.68%	19	6.76%	5	0.00%	0	1.35%	1	0.00%	0	0.00%	0	74

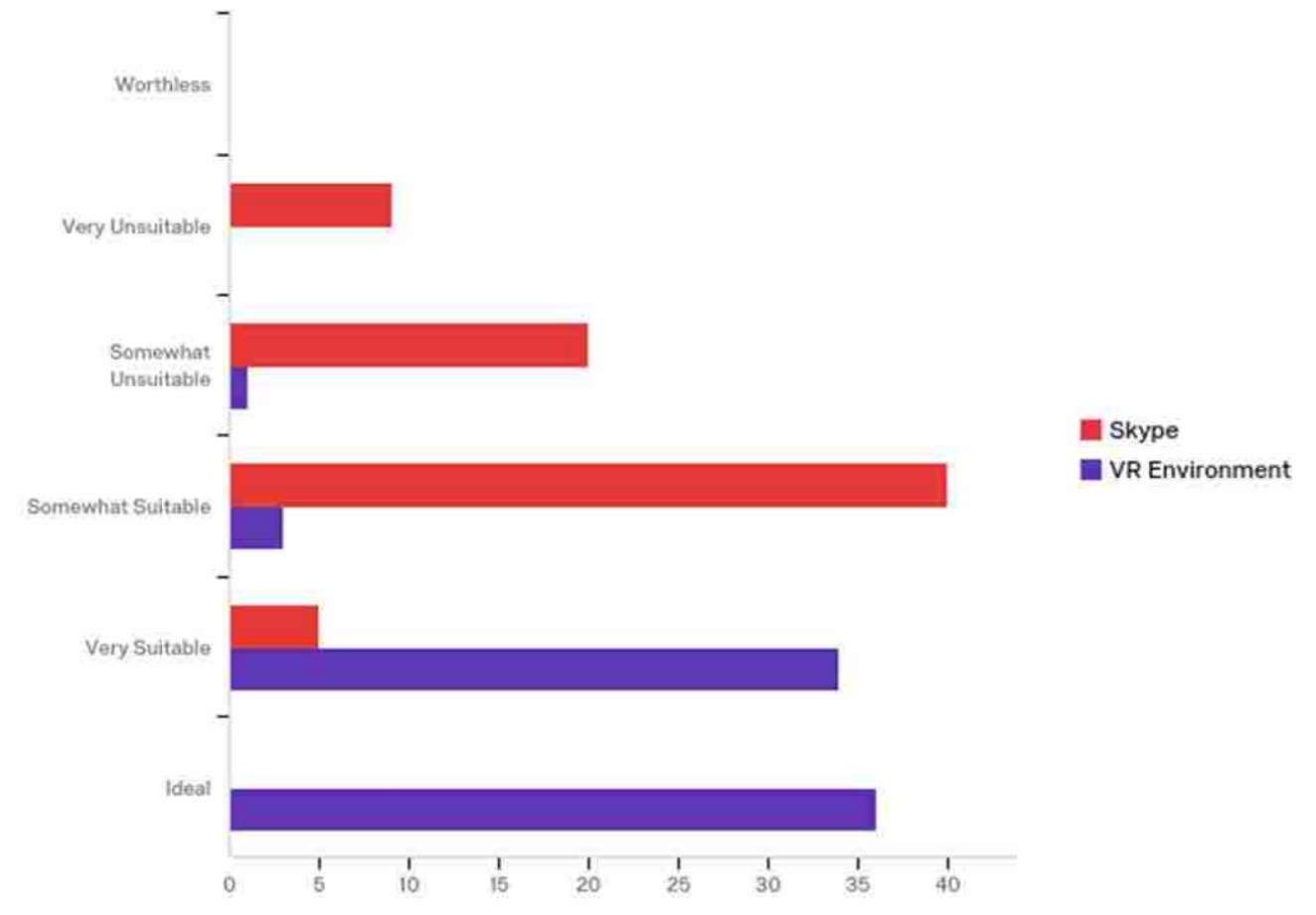
	engineering designs.															
6	I would like to have access to Virtual Reality tools in my future workplace.	59.46%	44	27.03%	20	10.81%	8	2.70%	2	0.00%	0	0.00%	0	0.00%	0	74

Q6 - Please rate how natural or unnatural the use of communication gestures (such as pointing, waving, "talking with your hands") felt in the VR Environment.



#	Answer	%	Count
1	Very Natural	20.27%	15
2	Natural	48.65%	36
3	Somewhat Natural	24.32%	18
4	Neither Natural nor Unnatural	2.70%	2
5	Somewhat Unnatural	4.05%	3
6	Unnatural	0.00%	0
7	Very Unnatural	0.00%	0
	Total	100%	74

Q1 - Please rank the suitability of each system you used for communicating complex 3D data (such as the task you performed).



#	Question	Worthless		Very Unsuitable		Somewhat Unsuitable		Somewhat Suitable		Very Suitable		Ideal		Total
1	Skype	0.00%	0	12.16%	9	27.03%	20	54.05%	40	6.76%	5	0.00%	0	74
2	VR Environment	0.00%	0	0.00%	0	1.35%	1	4.05%	3	45.95%	34	48.65%	36	74

Q8 - Please describe what you feel makes Skype "[QID1-ChoiceGroup-SelectedAnswers-1]" for communicating complex 3D data.

Please describe what you feel makes Skype "[QID1-ChoiceGroup-SelectedAnswer...
It's good in the fact that you can see the model (minus the path), and are still able to visually show the path, however the z-axis just isn't there, which makes it difficult in explaining depth. However, coupled with verbal explanation, the Skype environment still worked in getting the message across, although there was some imagination that had to be used due to the lack of z-axis.
I could still show my screen to another person and give them a rough idea of what it looked like from two different angles. I was not easy to describe when it came to a 3D object.
It allowed for me to show a general path, but unless the other person purposefully visualized it as well, it did not convey as strongly and as accurately as you could through other means.
With the screen share you can use your mouse pointer to indicate drawing, with the drawback that it is 2-D. It felt more natural to talk to the other person though.
The lack of ability to communicate what goes on between key points. It is possible to convey a few key points where significant changes occur but it would be very hard, if not impossible to show exactly what is happening between key points.
It's more difficult to give concrete measurements and angles
You're able to show what's going on with your mouse and stuff, but it's somewhat unsuitable because you're not able to visual three dimensional movement nearly as well as you are in VR.
It's just hard to communicate that sort of stuff when you're not in the same place.
I felt that the 3D model in skype was much more difficult to manipulate than the VR space.
Gestures and references are only made in 2D. 2D image of the computer monitor. Although there is depth to the images, the pointers provided only operated in 2 dimensions
Skype provides more than one angle to show the project. This makes it possible for the viewer to see more than his own angle of the environment and how to perform the task.
I feel like Skype is great for verbal things but a lot of they visual can get lost in the communication.
Screen sharing helps a lot, but talking about angles and start- and stop-points is hard to do via distance. Skype was okay but not ideal.
there's only 2 dimensions.
When we explained the path that was to be drawn in the VR environment through Skype we essentially had to describe it in 2D 2 separate times--once for the aerial view and once for the horizontal view. This was a little confusing and worked just fine, but the VR did it faster.
You have to do a lot of explaining and repeating of information in order to be sure both parties are understanding and communicating correctly. Easier for miscommunication. However, screensharing and 3D model makes it do-able and somewhat suitable.
It was to hard to explain depth and I could not see it from the point of view of my companion. The 2D design did not allow for in depth explanation
It was very difficult to understand depth in 3D space without any way of drawing the lines or paths. It

was nice to see the 3D model, but it was hard to know spacing
The Skype share screen feature was useful to chang orientations, and did allow for good communication as we could see the other screen and understand what was being communicated. The only problems were directions in multiple axis. When I had to communicate an angle or a change in the z direction it was difficult to completely show and ensure understanding in Skype.
You can see the person's face which helps with communication, and the mouse although not as ideal for showing, was more precise.
We were not able to draw using skype. In trying to duplicate a path, it was easier to see the path drawn before drawing it myself.
Being able to see the same thing that the other person sees as far as the model helped me understand. I also felt that it was a little more natural being able to see the other person over the skype call.
Skype was difficult to communicate, because there was little way to explain how a line moved from point a to b. Also, because it was 2D it was difficult to put together the different paths of the line.
I could manipulate the model in space and had a curser but could only show lengths in 2 dimensions.
I felt that it was hard to navigate around with mouse movements. Had I thought of using a pencil and paper, that would have been easier to show.
With Skype it's just 2D and you can't really understand gestures
Because it was not easy to explain the 3d model because you cannot see in 3D, and you can not show an exaple of it.
Without any capability to draw or demonstrate the image in space, I found it hard to communicate the shape. I lost the shape in my mind as I was trying to describe it without any visuals.
It was just a little harder as far as communicating with motion for exact points. Relied more on verbal instruction where as in VR its more easily shown.
You can't really show what you're trying to explain with much detail. Like trying to explain the distance each line had to go and how far down it traveled was very vague and not nearly as accurate as literally showing them with your hands as you could do in the 3d enviornment
We are able to see and hear which makes explanation possible, the screen share helped too. It was difficult though because you could not see the 3d aspects of what they were describing.
It's nice to see the model in screen share, but it's not as fast or easy to visualize as seeing someone model it in VR.
While it was useful to be able to share my screenwith him and rotate it around, it really wasn't very convenient if I needed to explain any sort of depth from my model too him. If I had the ability to highlight specific portion or mark points on my model then it would be much better but as it was I spent more time try to get the right viewing angle than explaining the line he was supposed to trace.
With complex 3D data, it is difficult to convey that information outside of virtual reality as the other person simply can't visualize what the other is trying to draw/image.
I use skype all the time so I am comfortable explaining and showing things. I believe it is useful to show someone something, but not as valuable as in the VR environment.
It involved a layer of translation; I had to use words and gesture with a mouse to communicate everything.
I was able to rotate the image and use the mouse to better show where in space i was gesturing to. the mouse was a lot more sensitive and accurate than the gestures in VR.
I could get the point across for the more simple shapes that were in one plane, but it was much easier to explain the more 3D lines via the VE.

It was hard to see the third dimension of what he was describing, I only could gather from what he explained

It was very hard to express a 3d model over skype even with sharing your screen. If i could have drawn it in the cad program that would have been different

It was difficult to describe a very specific visual using words or "almost" visual effects.

I felt that I was able to get my point across, but it wasn't natural and the limitations of skype got in the way of being able to communicate in a simple way.

It's really hard to visualize the 3D paths when you don't really have anything that keeps track of it for you. It's hard to represent a 3D object, in a 2D screen projection

It's difficult to get a good idea of motion in planes perpendicular to the motion, and we are facing each other instead of looking at the same model.

it was hard to show the path on the machine arm, it was hard to show depth so you had to show the path from a few different angles

You can see what the person is talking about through hand gestures and whatnot, however, you can't tell depth at all. It was hard to tell where things were in a 3D environment.

I think it would depend on the program you used, but trying to explain a 3D model in two dimensions, has the same limitations as paper. It is not any better than a picture.

Easier to rotate and point to paths along the model. Easier to pause for explanation.

They say a picture is worth a thousand words. In the Skype environment it was very difficult to communicate in words what the path looked like because it severely lacked in visual aid. I did not mark it as "worthless" because you could still communicate the path, and gesturing with the mouse helped to some extent, but it was very difficult to communicate.

It was sometimes hard to understand what the other person was trying to explain because I couldn't actually track what they were drawing and the 3D part made it harder to explain verbally.

Difficult to describe orientation of the object. Took a lot of time to describe .

I am a virtual person, so I thought it was a lot easier to understand how to draw something based on my partner drawing it on the virtual environment. So, Skype can work, but only if the person is very good at describing what I am supposed to draw out.

there's no way to have parties moving their hands in a 3d space to compare the relation one fels to the 3d structure.

In 3D space there is really no better way to communicate than with an actual 3D environment. Skype just does not provide the same experience that a 3D environment would.

It's somewhat suitable because you can still get most of what you want to get across because it's 3d,, but you can't draw with it. it's better than having to show a 2d drawing over the camera, but yet not as good as drawing it in a 3d environment.

It was convenient to transfer the screen to the other person, but it would have been much easier if we could have drawn lines.

It was really easy to look at the same model via screen share and understand the instructions.

Using a 2 dimensional environment to communicate 3 dimensional data was difficult, especially after recently using VR and comparing Skype against that.

Although you are describing whats happening and you are talking with skype, there is still an unclear/ambiguous aspect to it all.

the ability to visually see each other, and talking felt much more natural

It is difficult to show the exact points of where objects end and begin.

You can show the screen of an object preview, but it's really hard to show the path of travel

I feel that a important part of communicating/ describing a physical component requires the ability to see depth as well as multiple views. Skype did not seem to have a very good depth feature. In the skype model the "Student" was not able to manipulate the model, this made the learning and comprehension more difficult.

Having to shift view angles in order to explain depth is not ideal and distracting.

It was nice that you could rotate the model and point with the mouse, but it would have been better if you could actually draw a line and give them a clearer visual.

Its alright, but manipulating the angle and then tracing out is not ideal and doesnt allow for 3d descriptions.

You are able to orient the 3D model screen share in whatever orientation was necessary to clarify what direction and place each line and point were going. You could easily turn the 3D into 2D to get more specific directions.

I was able to share my screen with my partner showing the 3D model and showing different angles while dragging my mouse to try and explain what the path looked like. The problem comes when trying to explain a 3D model using 2D examples.

The communication is good if talking to someone far away but a 3D model is the way to go to explicitly explain how to perform tasks

You are really only limited to two types of communication when it comes to skype. You can show your companion the overall basis model while you converse with them while in the virtual reality you are able to show them the table you are working with, show them the design path and talk with them about what you are working on.

It was easy to have things not moving as much and be able to use my mouse.

The idea can be put across but there is still a lot of room for error

It requires the "student" to create 3d images from the 2d snapshots they are given, and even simple 3d designs can be easily mis-interpreted when made 3d by someone other than the designer.

hand motions and screen sharing available to communicate

Q9 - Please describe what you feel makes the Virtual Environment "[QID1-ChoiceGroup-SelectedAnswers-2]" for communicating complex 3D data.

Please describe what you feel makes the Virtual Environment "[QID1-ChoiceGr...
In the VR environment, the teacher was able to actually draw the path, if necessary, showing firsthand what was supposed to take place. The ability to walk around the model was very helpful as well, giving the student multiple viewpoints in which he or she could gain a more complete view of the task at hand.
I was able to draw a 3D image of what I was trying to communicate. They could see it from my perspective and it was a lot easier to describe.
I liked this way much more than Skype because you could actually see what was going on. Also, it didn't have to be super accurate but even just by showing certain points on the actual objects, I had a much easier time visualizing what I had to draw. I could more quickly see all 3D aspects of what I had to draw. Also it allows for more than one type of learning style to be exhibited.
You can walk around the model and see it from any angle you want. The learner is more active because they can draw their own lines and look from whatever angle they want.
The ability to see the actual shape being described.
You can trace exact curves, angles, and lengths. It would be ideal if it were easy to draw a straight line, because hand trembling throws it off slightly.
You can point and move and recreate three dimensional movement exactly as it would be in the real world.
You can see exactly what is being drawn and it's not a complicated interface to use. You just have to move your body like in real life to see from a different angle
It was easier to follow gestures and understand where you were supposed to draw in space.
3 dimensional accuracy
In the environment, it was quite interactive, using visuals such as drawing or pointing.
In VR, it is a lot easier to visual the 3D environment and convey the meaning by showing.
I don't feel that the VR environment is optimal for explaining 3D data, but it was certainly better than using Skype. Using 3D to show 3D data makes sense. VR is not ideal because it's not a visual of an actual 3D model. It's only virtual 3D.
it's hard because we're drawing with our fingers, but it's really good to be able to change perspective and use three dimensions easily.
The drawing was a little finicky which made this not "Ideal" but it was very good for the purpose it was made for. I would have no problems using that more.
Words are much less necessary, simply show them and they will know the degree to which things curve etc. without so much possible miscommunication.
It would even be better than one on one real life interaction, because it allows for the creation of certain 3D shapes and designs and we did not have to be in the same area to do so.
It was nice to physically move myself around the model and see the model from a different perspective without making others change theirs.

In VR, I could physically see and copy exactly what my partner was showing. There was little to no room for error as I could even practice creating the 3-D image while being instructed. We could see the exact same perspective, the same changes in 3 dimensions, and have no miscommunications about depth or structure.

It was really great to be able to both work on the model together. And to be able to actually draw the pathway and provide clarifications.

it was kind of difficult to draw with my hands but i much preferred the VR environment's ability to draw and communicate.

Seeing the exact motion of the other person has he drew the figure helped me understand what he was trying to convey.

It was easy to show and learn exactly how the line moved where it moved to and it was a lot easier to visualize.

I was able to understand how the path interacted in 3d space. It was very intuitive to understand the 3D models.

It was time consuming and a little exhausting to try and complete the tasks. I suppose this could work very well, depending on what the task was.

You can literally see from the other person's perspective

you are able to show what you are trying to teach, you can not just see it , but you are able to interact. It was really useful.

I could clearly see the shape, and follow exactly what I was being taught. It was very easy to understand how the shape looked and felt in real space.

You can indicate very clearly through motion what is going on in real time where more verbal instruction would be used in skype because you couldnt draw with that particular program.

You can easily show what you're trying to do with your hands on the actual object. There isn't much guesswork and it's just following the other person's hands. You can be much more specific

Being able to actually see them draw what they are trying to explain helps you really have a visual in your own mind rather than having to create a visual from description.

It's perfect because in VR you're in a 3D atmosphere whereas with skype you're dealing with 2D or 3D that you can't interact with.

It was very convenient that I could rotate around the model and he could as well to see the different directions I took when drawing. It was also tremendously useful that he could see and also put his input into my model. It allowed us to show exactly what we wanted each other to do and clarify it really easily after if there was any miscommunication.

With the virtual environment, one may be able to draw out what they wish to convey, be it an architectural design, or a simple framework for a conceptual design. It eases the ability to communicate complex 3D data as a whole.

It is 3 Dimensional. I am a very visual learner and you can just show the other person rather than explaining it.

In VR, I was able to point and gesture naturally, while augmenting it with my words and gestures.

Though the gestures were a little sluggish and inaccurate, i felt i was better able to show exactly where and what shape the lines should be

As long as I knew my partner knew what he was talking about, it was really easy to follow the drawings he made and the instructions he gave me. Likewise for teaching, it was much easier just to draw the shape for him.

You could see the 3D element, and he could draw the model. Seeing was nice.
I was able to gesture, draw guide lines ect
The real-life image we were trying to recreate in virtual reality was easiest to recreate using something you could see. Between the virtual environment and the skype, the virtual environment was more effective. However, if you were using something like pipe-cleaners to show the lines, I think it could have been almost as effective using Skype.
It felt natural. I could use my hand to signify where I was talking about, and virtually "show" my partner what I was thinking in 3-dimensional space.
It felt very natural, and I was able to talk and use my hand gestures to refer to the model as well as drawing lines in 3D space which made describing a 3D object seem very natural
It was great visualization, I could move and view the workspace form multiple directions, and it was easier to distinguish motion in all three planes.
with the VR I could see the depth better than with the skype and I could show where certain points were and see that my partner basically knew where I was referring to
You can see exactly where something was supposed to be and understand how things are supposed to come together.
Models are much easier to understand than pictures. It is easier to understand how 3D fits together
I had problems drawing continuous lines and "pinching" to erase the lines drawn. It's like I have to relearn how to use my hands.
I was able to draw exactly what I thought it should look like, and communicating using hand gestures also helped immensely. As someone who has interned with a product design firm I can see how using VR to communicate complex 3D models would be extremely useful.
It made it a lot easier to communicate and explain exactly what you were trying to draw. The hard part was just that I didn't feel comfortable drawing but with experience I can definitely see how it will make explaining and creating projects a lot easier.
Both teacher and learner can view form the same point easily. It is more intuitive.
Virtual Environment really helped me understand how to draw the lines and what planes the lines were in. Because I am not an engineer major, it was very helpful for my partner to just draw it out for me rather than explain it to me.
it allows to people to look at a item in a 3d space much as you do when you are both present in room.
It is a 3d environment capable of helping you see the actual picture instead of having to use 3 different sides of the same thing to describe a certain point, you can just point at it and or draw it.
You are able to actually show with your hands the path you were trying to make, and even draw when needed, and you have the 3d environment to help show it.
It was easy to just sketch something out in 3D that the other person could see and then follow. There were some bugs, but as the technology improves I'm sure those will be worked out.
The best part of this was the fact that you could literally draw what you wanted and the person could see it in 3D and get a perfectly clear idea of where it should be.
A 3D environment for 3D data just made sense, it was nice to be able to see what the other person was doing along with being able to talk about it.
because its immersive. I think it's a lot easier to understand the data when you can actualy see it and manipulate it

I could describe both verbally, and draw out the shapes. I could point out things like turns, and rates of elevation change much faster. I also felt that percise placement especially with distances was better.

You can draw the object right there for them to see. Plus you can mentor their attempts and give live feedback.

You can draw what you mean so others can see exactly what you mean.

The virtual environment allowed both people to manipulate the model, and view the model at different angles. The two participants were able to see each others hands and see where they were going with a idea, gesture or action.

The ability to freely look at a design from any angle made the experience feel very natural. By getting more into the digital world helped me feel less in it.

It was better that skype in that you were able to actually draw a line for the other person to see, however it was not ideal because of the limitations of the technology to accurately capture what you were gesturing at. But with improvements to the system I think it could be ideal.

It makes it possible in one movement to express a complex 3d direction or idea.

It was "Draw a line from here to here, then here to here" and so on. When teaching you could plot the points where the lines connected, draw example lines between them to show any curvature, and they pretty much had the figure drawn for them.

Virtual reality allows for all different views on an object which facilitates picturing it in one's mind in a real world situation.

With the 3D interface, I can perfectly point out where to start and what points to look/draw at whereas a 2D explanation could be difficult to explain on a 3D plane.

It's ideal because you are able to show them exactly what they need to see. If you are going to show them a design path, you can draw that exact path. It was almost awkward how much I was explaining what I was intending to show.

You are actually able to see what the design looks like, it just saved a lot of time to try explaining the different designs.

A person can literally draw what they're talking about and the other person can see it as if they're right next to them.

I can basically make a 3d, virtual prototype shape instantly.

easy to draw and see things in 3D

Q19 - What would you like to see changed, improved, or added to the Virtual Environment if anything?

What would you like to see changed, improved, or added to the Virtual Envir...
Maybe little puppies to pet.
I would like to see a way to erase portions without having to get rid of a whole segment. I usually liked muy lines but not the start or the finish so I would like some way to "clean it up." I was also hard to start/stop lines accurately.
Try to make the pointing more natural. It took me a while to get used to having my hand open and then close with only the pointer finger still showing to draw. I'm used to the other way with having my hand closed first and then opening my pointer finger to draw. The pinching was also somewhat unnatural just because it got confusing as to which finger needed to be closed. Maybe having a different motion like having the whole hand open and close (kinda like a clasp) with both hands that would be easier. I did like the motion for erasing everything though.
The whole system was a little shaky at times. It would be nice if I could freeze the camera angle better than just holding my head still. Also holding my finger up to draw for extended periods of time made it hurt a little.
The ability to delete parts of lines that are drawn, perhaps with another hand gesture. Also possibly a haptic type interface that will give a force feedback when touching an object or a line drawn in the virtual space.
A way to delete small portions of line and a way to split a segment into multiple pieces (with a karate chop?)
Nothing.
Better graphics.
It was sometimes hard to manipulate where/when you were drawing.
More tools i.e. ability to move drawn objects, create other objects besides lines such as cubes or spheres.
If Skype and VR can be merged, allowing the individual using the VR to draw on the skype screen and having the partner see and interact with however the "screen master" chose to layout the current view of the project.
An easier way to delete? I found it somewhat difficult to easily select and delete portions.
In the case of this experiment (this data that we were communicating), it would have been nice if the fingers could have been assumed to be drawing straight lines. The finger movements (including drawing) were so inexact that it was distracting. Maybe there could be a straight-line option when the angles and distances were important and when the line was supposed to be straight.
to be able to use some sort of drawing tools where we don't have to draw untidy lines with our fingers. Also the virtual hands didn't always respond to my hands properly.
More responsive drawing that eliminates hand errors causing little lines to show up. That was frustrating. Also about 20% of the time (maybe less) the line would stop drawing immediately after starting to draw, so I would have to close to a fist and point my finger again.

Wireless headsets.

The sensitivity to deleting lines and drawing line so that small mistakes are not made.

Improved graphics would be nice. I think it would also be better to have sensors that are more accurate to have fewer paths drawn by mistake

There were a few things regarding the draw features that glitched out at times, but that's simply a development issue. Tracking my hands to know when I want to draw and when I don't was a temperamental issue. I have no solution for the issue, but if there were a way to see when I leave a plane or axis, it would be easier to sketch my drawings that I want to only follow a 2 dimensional path. There are times when, by using my arms, I was unable to keep my hand the exact same distance from my body while drawing a line, and instead of simply drawing a line straight down, I inadvertently drew a line that progressed down and out and away from me, along the z-axis. If there were a control feature that could be implemented to ensure that I only draw within one axis when I designate it as such, it would help with my accuracy.

The ability to draw straight lines and curves. More accurate reading of hand gestures.

maybe it would help to use tools to draw, like if you were to pick up a pencil and draw. That way you wouldn't draw so many unwanted lines and things by accident when gesturing.

How jumpy my hands were. I felt like i had to redraw a lot of things and also reprompt my hands so that i could begin drawing.

A way to draw with straight line or curved line more like a real cad program, but you can use your hands to adjust the line to be curved.

The ability to manipulate the base model in order to minimize user movement (such as going down on one knee)

I think it would be easier to use a remote rather than your finger. (The accuracy of pinching and drawing with the finger was not as good as I would have liked.)

not yet

maybe a better way to control when to draw or not, that you can avoid accidently drawing something you don't want to

I thought the VR Environment was lovely. It was an open space free from distractions.

Just the control fluidity which would take time and aren't horribly in the way just take some getting used to .

The only improvement I could see is that it recognizes the hand gestures better. sometimes trying to erase was difficult as the camera read my fingers wrong. maybe adding a video game controller for more precise movement would make it more effective

The hand recognition struggles a little bit but I wouldn't know any suggestions to improve it.

Improved gesture recognition and graphics.

I've used an HTC VIVE (albeit once) in the past and at least for the experiment we did I think it would be much easier if you used the controller in this. My hands glitching out or drawing where I didn't want to could be easily fixed with with the controllers for precision. It would also be very useful if I could mark points on the model, "Draw your line from here, to here." Sort of thing. It would give even greater precision for the lines drawn I would say.

I would strongly suggest updating the system's ability to track finger motions (perhaps utilize a glove of some sort for better accuracy?). This was because the system had difficulty tracking what I wanted to

draw out.

I really thought it was great. The ability to create sharp and straight lines by pressing two points in space would be nice. The hand motions took a little bit of time to get used to.

I found that it was tricky to draw paths straight. I also wish there was something physical I could hold onto.

Better accuracy of the virtual gestures. I felt it didnt accurately show or do what my hands were doing.

The headset did a great job reading my hands, but was sometimes inaccurate with more complicated hand gestures. Otherwise very good.

A drag and drop tool, like in paint or word, to have better control over the drawing.

more sensors to watch hands and the whole body

I am a pretty tactile learner; I like using pens and paper or actual arts and crafts. The virtual environment felt foreign to me because I couldn't grab onto anything. Maybe if I had been trained using an actual string or something physical to learn, my brain could connect to the virtual environment better.

I would prefer to use a different sign for drawing, because I wished I could use my finger to point at things. It would have felt more natural if the other person had a body in virtual space, even if it was partially translucent.

I did feel that the pinching gesture to erase was a little awkward. This was my first experience with VR, and it felt very natural; I was able to catch on petty quick.

The drawing was sometimes difficult. I would like a more consistent way to distinguish between motioning and drawing.

the hardest thing was getting the hands to recognize what you were doing so maybe using a controller would help to be a little more responsive with the commands.

If it were wireless, that would be awesome. I felt like the cord was a bit distracting.

Tools would be nice, rulers, protractors, etc. Also the ability to change our view without moving, such as being able to zoom in/out

Smoother movement, ability to make smaller curves.

The sensor camera could be improved to make drawing and erasing easier.

The sensors was the only thing that I think could be improved. It sometimes had trouble picking up my hand gestures but that could have been hand error placement on my part too.

Tools involve into it, that can make it more precise like a virtual ruler or something else

It was hard to have my hands draw lines or objects at time, perhaps due to the position of my hands or the sensors. It was the better method, but it was a little awkward to draw at times.

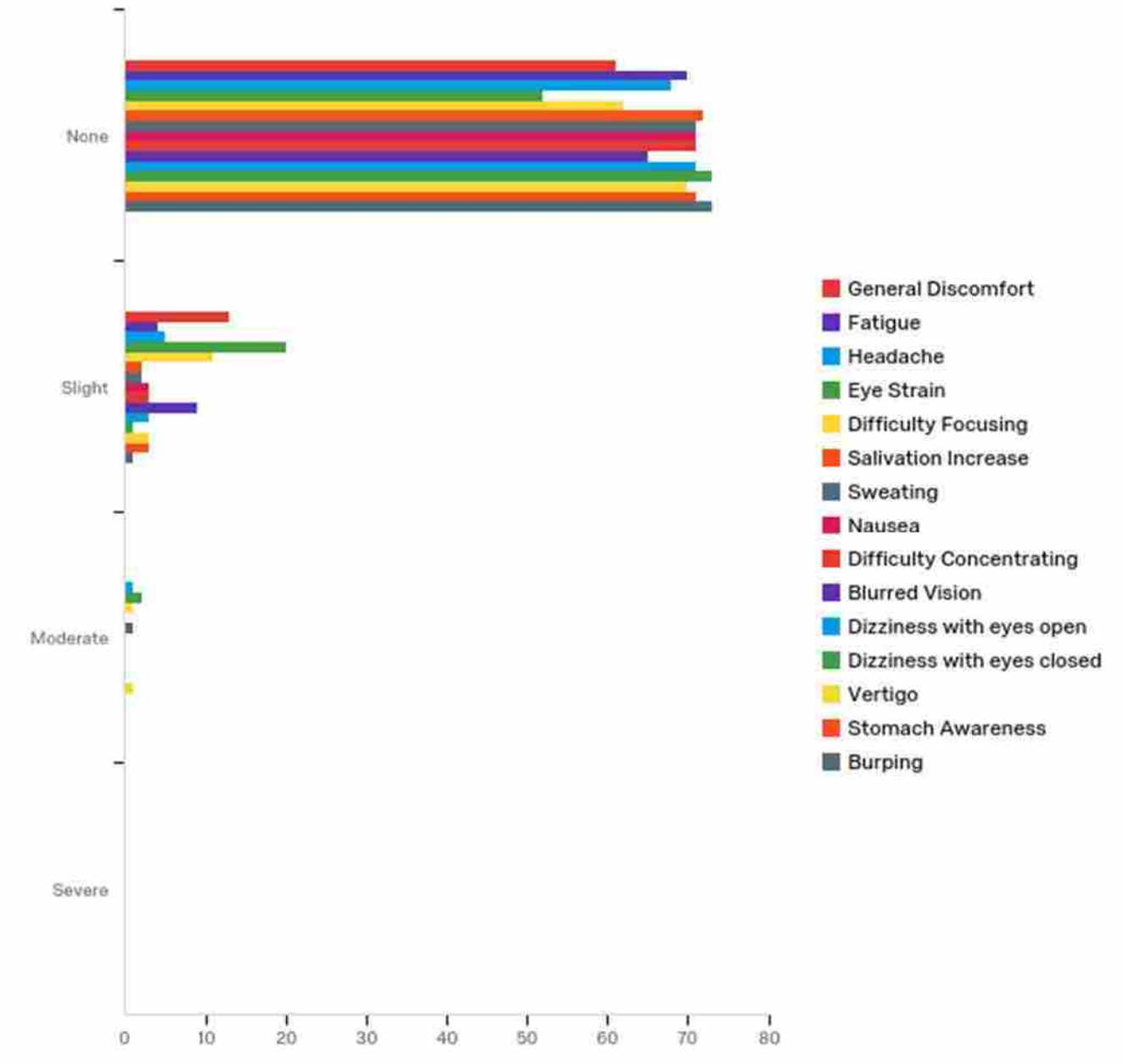
the ability to masher i think would improve the ability to compare and realt a structure. i also think that the ability to import simple references shapes would be helpfull (ex squares, circles, cylinders). and a prospective channing setting. for instance the item we have in reality is much smaller than the one in vr, it would be nice to zoom in and out.

the only thing that took a little getting used to was the hand movements to interact with the environment. The rest worked just fine especially when pointing to things or gesturing.

just improvements with hand tracking and when you actually point to draw or doing thumbs up to erase everything. I felt it wouldn't do it every time, but I was still able to use the jestures. possibly also be able to move the board around, if possible.

Just making it more reliable to see my hand gestures. It often was confused with what i was doing, especially when I was trying to pinch and erase lines.
The drawing command was kinda glitchy and that made it difficult to draw exactly what you wanted.
More accurate readings of hand gestures.
The option to add extra colors, or possible present a static image would be very useful. There were minor glitching with hands, not being read.
Better hand tracking and recognition of drawing and erasing specific elements.
Better hand control. Rulers, and other tools so you can draw straight lines. A way to draw dotted lines.
I would like additional features such as selecting components, zoom in and zoom out, and straight line features.
Better hand tracking and less cables.
Better and more accurate sensors that more accurately follow the movement of your hand.
everything was fine, the pinching gesture was a little tricky at times.
Maybe an ability to draw a straight line of any length. Like put the fingers on your opposite hands together, and then spread them apart and a straight line would be drawn between them.
I've used controllers in the past which seem to be easier just for making smooth lines I guess, but the motion sensing camera using my hands worked as well.
The virtual environment was great. Only improvement I would suggest is better sensors so it would function more perfectly.
Perhaps some drawing tools, like Photoshop, where you can draw and snap perfect lines and curves.
tools, like the ability to make a straight line between two points and curved lines, because it's hard to make it exact with just the movement of my finger.
Somehow make the sensors that detect your fingers a little better. Maybe there would be some kind of glove to wear.
CAD implementation - create straight lines, define points, move lines, etc. All of that. Mostly the ability to move lines once they are drawn.
maybe tools to add a straight line or angles

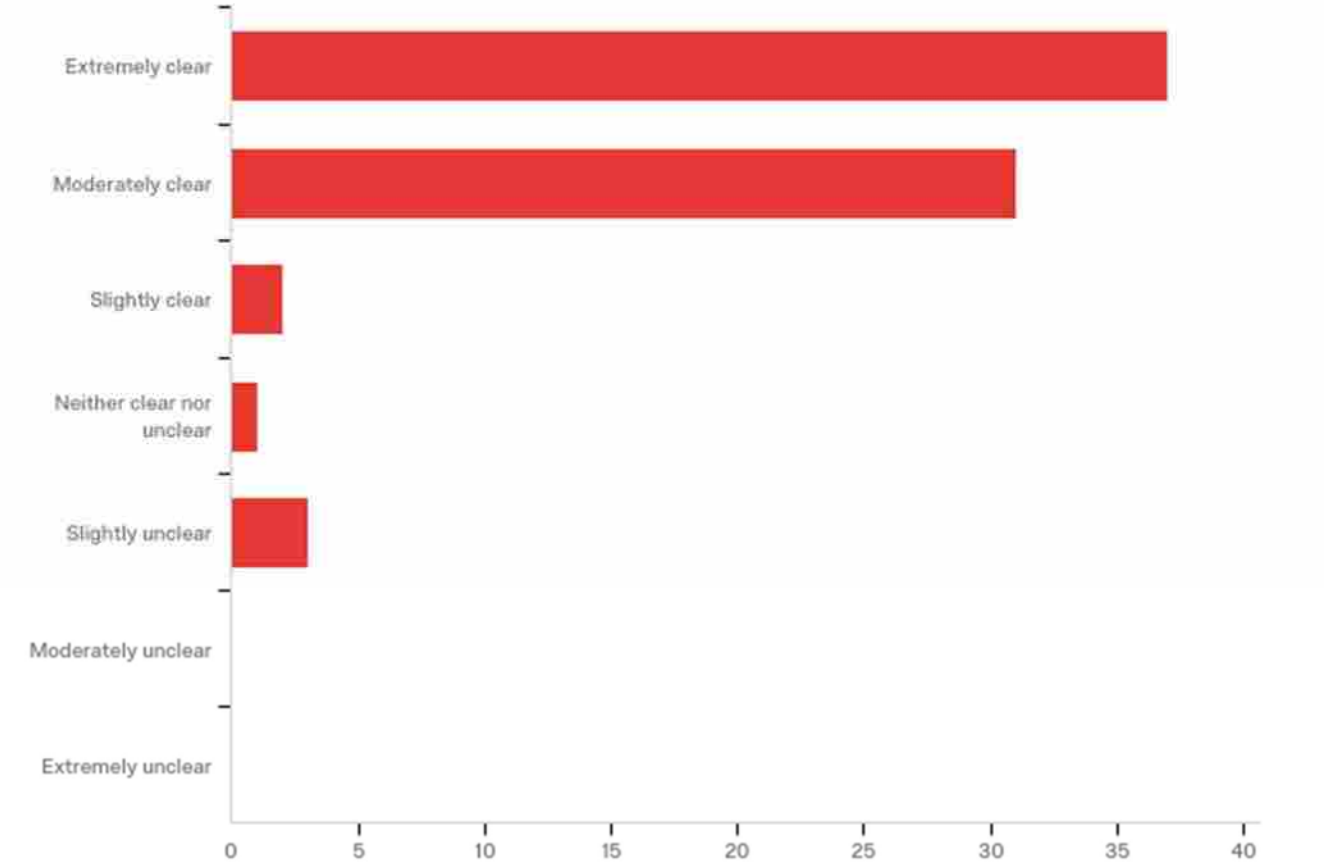
Q3 - To what extent did you experience any of the following cybersickness symptoms during or after using the VR Environment?



#	Question	None		Slight		Moderate		Severe		Total
1	General Discomfort	82.43%	61	17.57%	13	0.00%	0	0.00%	0	74
2	Fatigue	94.59%	70	5.41%	4	0.00%	0	0.00%	0	74

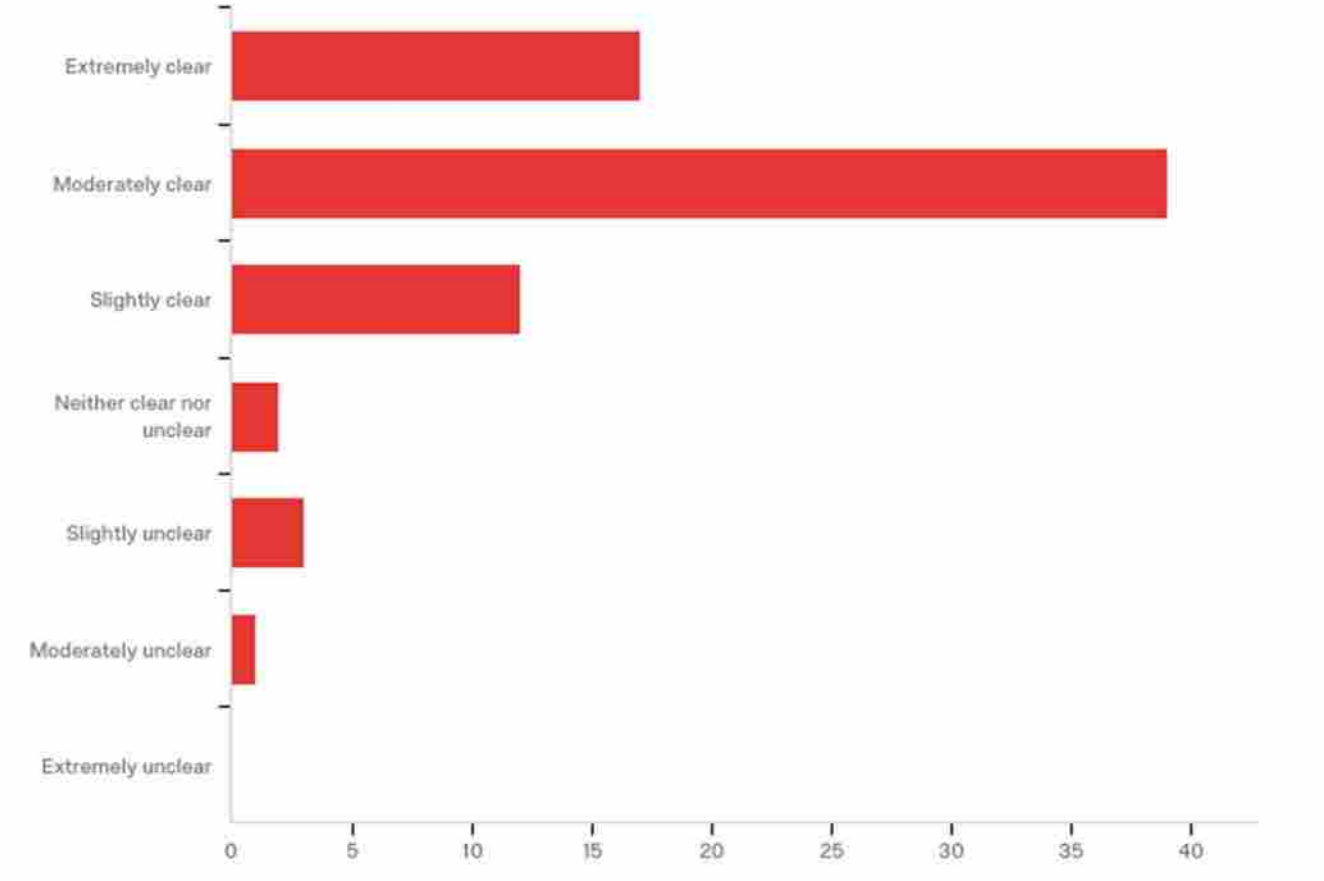
3	Headache	91.89%	68	6.76%	5	1.35%	1	0.00%	0	74
4	Eye Strain	70.27%	52	27.03%	20	2.70%	2	0.00%	0	74
5	Difficulty Focusing	83.78%	62	14.86%	11	1.35%	1	0.00%	0	74
6	Salivation Increase	97.30%	72	2.70%	2	0.00%	0	0.00%	0	74
7	Sweating	95.95%	71	2.70%	2	1.35%	1	0.00%	0	74
8	Nausea	95.95%	71	4.05%	3	0.00%	0	0.00%	0	74
9	Difficulty Concentrating	95.95%	71	4.05%	3	0.00%	0	0.00%	0	74
10	Blurred Vision	87.84%	65	12.16%	9	0.00%	0	0.00%	0	74
11	Dizziness with eyes open	95.95%	71	4.05%	3	0.00%	0	0.00%	0	74
12	Dizziness with eyes closed	98.65%	73	1.35%	1	0.00%	0	0.00%	0	74
13	Vertigo	94.59%	70	4.05%	3	1.35%	1	0.00%	0	74
14	Stomach Awareness	95.95%	71	4.05%	3	0.00%	0	0.00%	0	74
15	Burping	98.65%	73	1.35%	1	0.00%	0	0.00%	0	74

Q11 - Please rate how articulate your partner was when explaining the path to you.



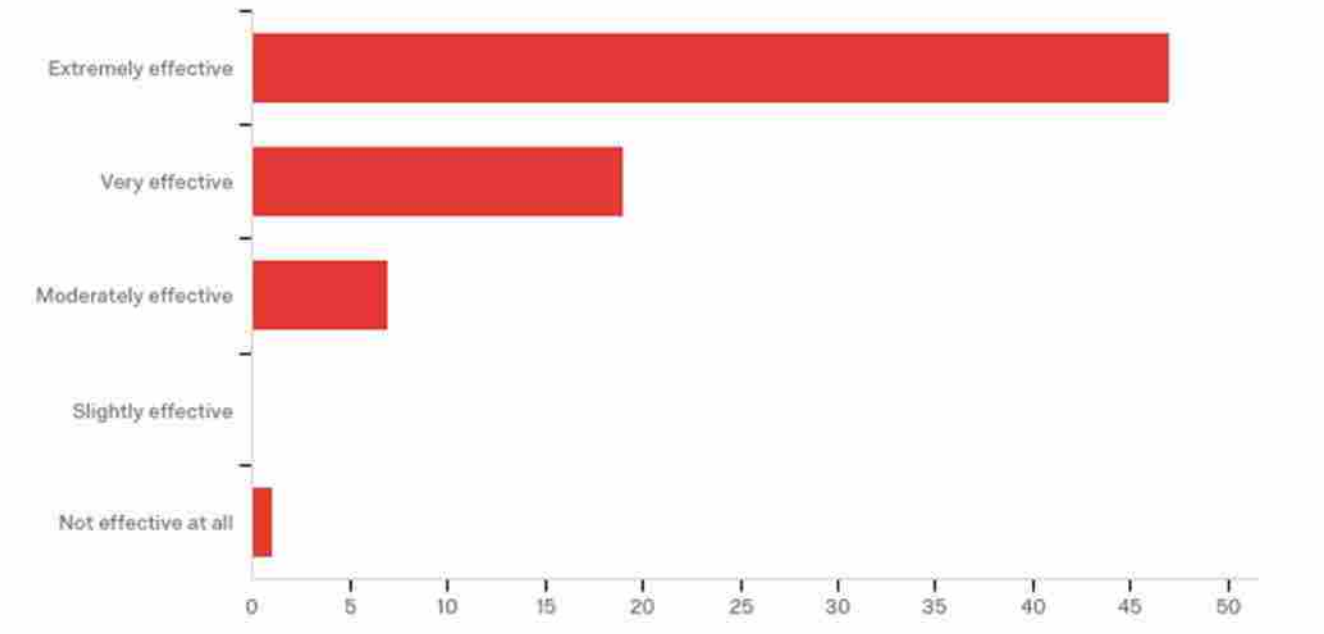
#	Answer	%	Count
1	Extremely clear	50.00%	37
2	Moderately clear	41.89%	31
3	Slightly clear	2.70%	2
4	Neither clear nor unclear	1.35%	1
5	Slightly unclear	4.05%	3
6	Moderately unclear	0.00%	0
7	Extremely unclear	0.00%	0
	Total	100%	74

Q16 - Please rate how articulate you were when explaining the path to your partner.



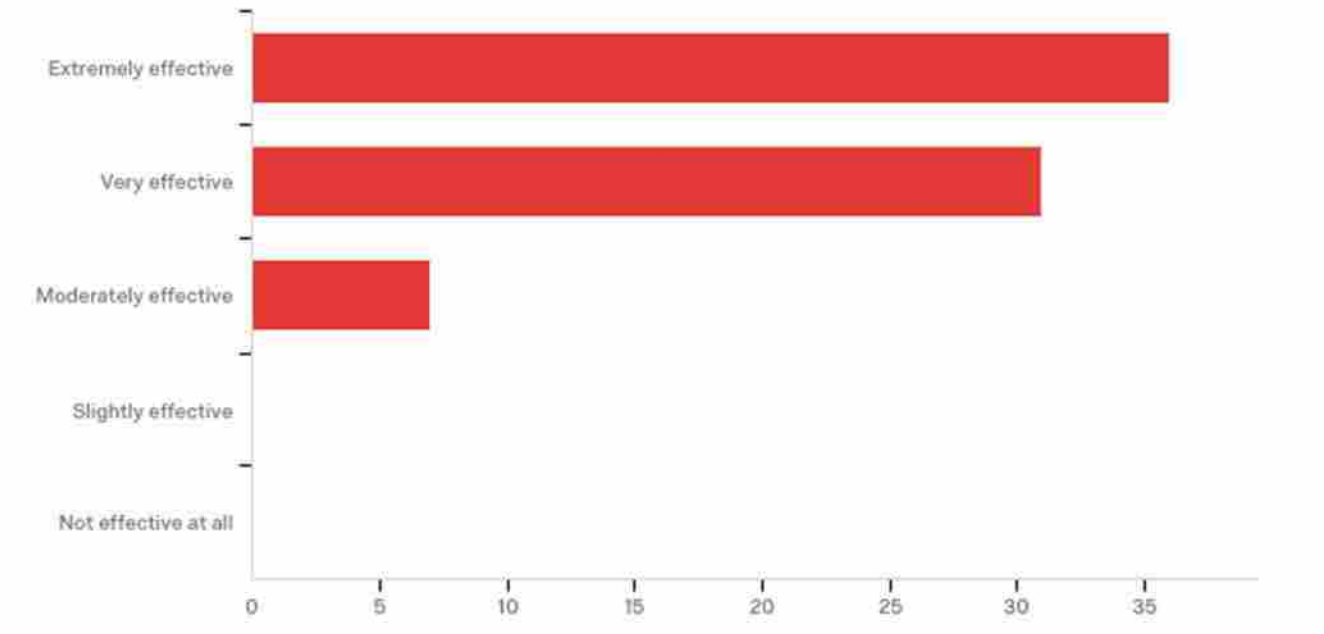
#	Answer	%	Count
1	Extremely clear	22.97%	17
2	Moderately clear	52.70%	39
3	Slightly clear	16.22%	12
4	Neither clear nor unclear	2.70%	2
5	Slightly unclear	4.05%	3
6	Moderately unclear	1.35%	1
7	Extremely unclear	0.00%	0
	Total	100%	74

Q12 - Please rate how effective your partner's hand gestures in the VR Environment were in communicating the path to you.



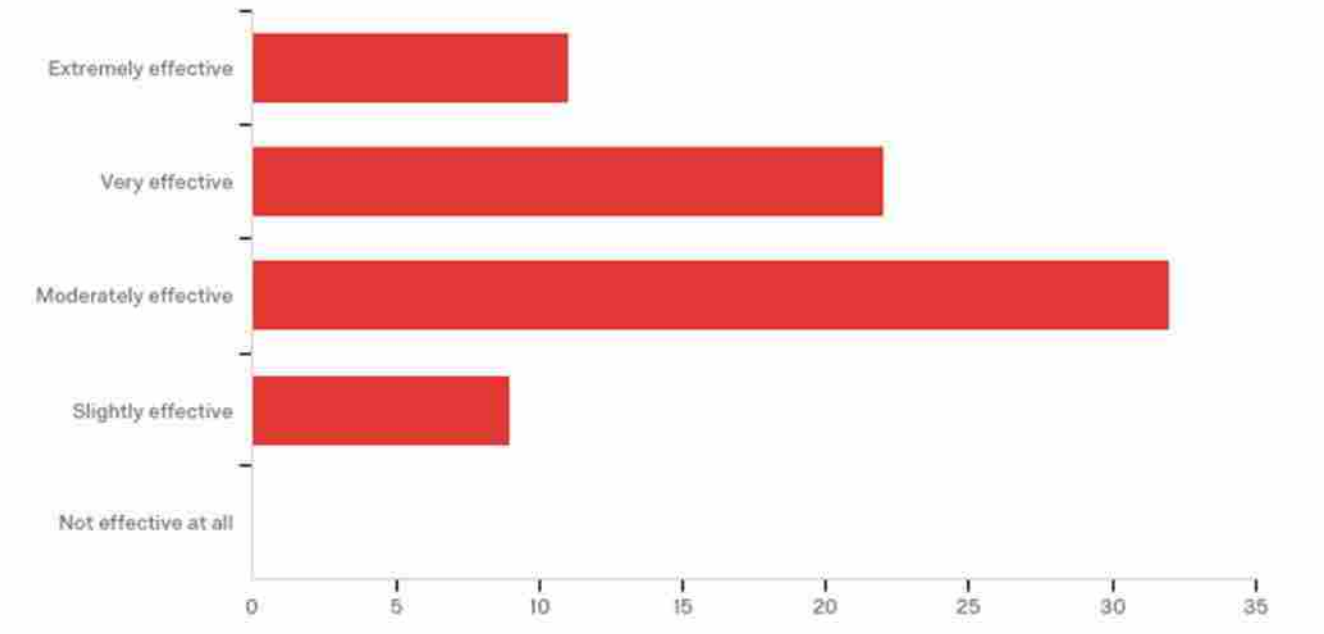
#	Answer	%	Count
1	Extremely effective	63.51%	47
2	Very effective	25.68%	19
3	Moderately effective	9.46%	7
4	Slightly effective	0.00%	0
5	Not effective at all	1.35%	1
	Total	100%	74

Q17 - Please rate how effective your hand gestures in the VR Environment were in communicating the path to your partner.



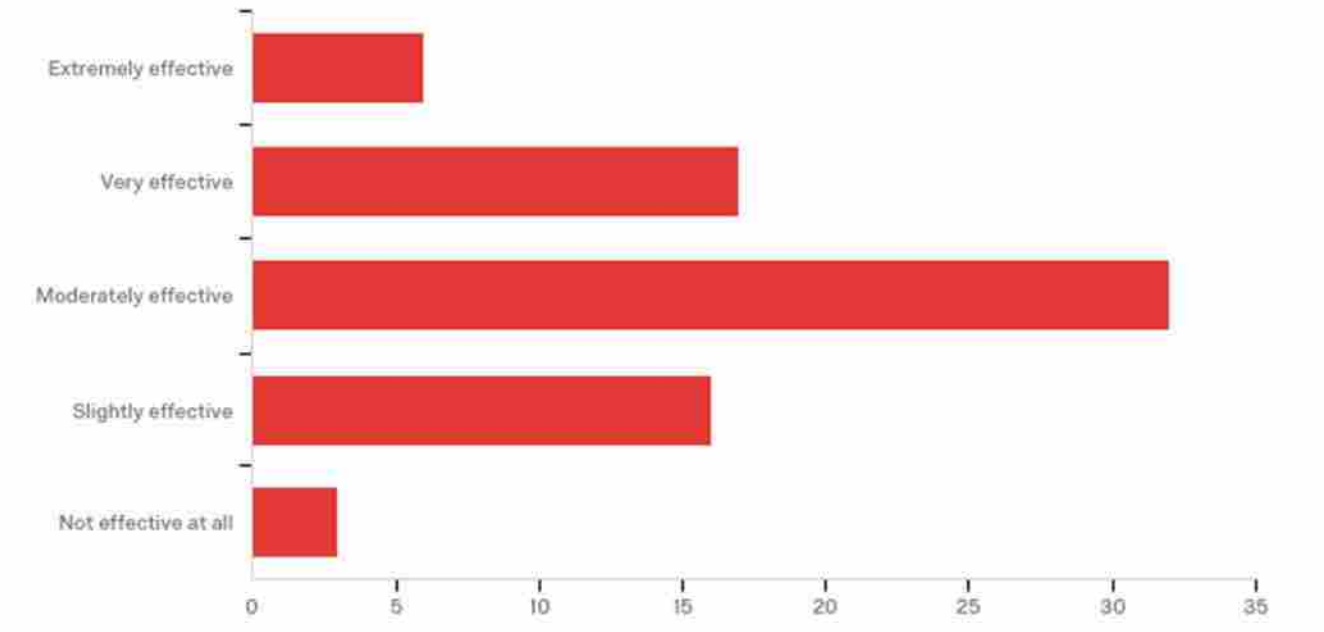
#	Answer	%	Count
1	Extremely effective	48.65%	36
2	Very effective	41.89%	31
3	Moderately effective	9.46%	7
4	Slightly effective	0.00%	0
5	Not effective at all	0.00%	0
	Total	100%	74

Q13 - Please rate how effective your partner's mouse gestures in Skype were in communicating the path to you.



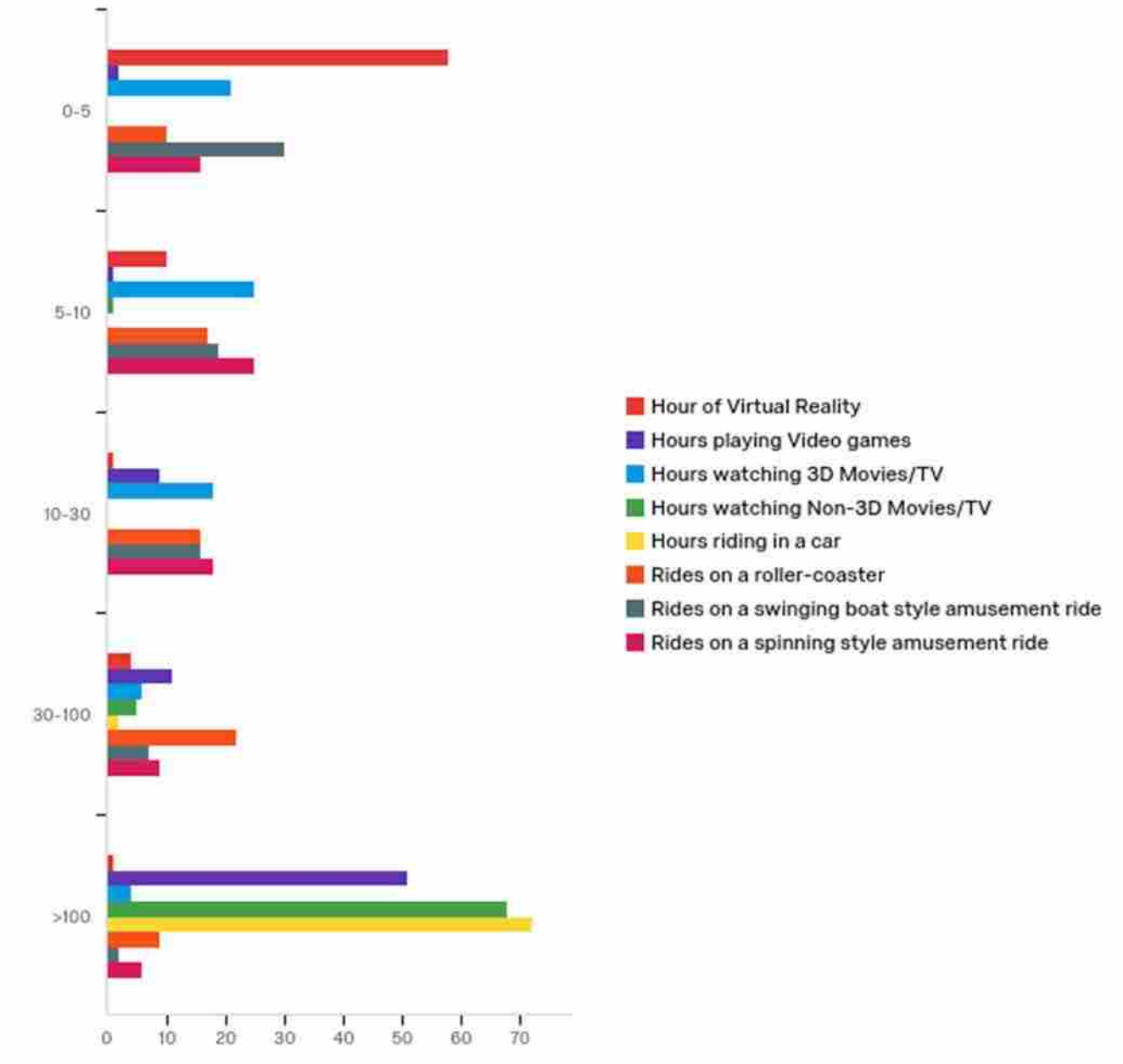
#	Answer	%	Count
1	Extremely effective	14.86%	11
2	Very effective	29.73%	22
3	Moderately effective	43.24%	32
4	Slightly effective	12.16%	9
5	Not effective at all	0.00%	0
	Total	100%	74

Q18 - Please rate how effective your mouse gestures in Skype were in communicating the path to your partner.



#	Answer	%	Count
1	Extremely effective	8.11%	6
2	Very effective	22.97%	17
3	Moderately effective	43.24%	32
4	Slightly effective	21.62%	16
5	Not effective at all	4.05%	3
	Total	100%	74

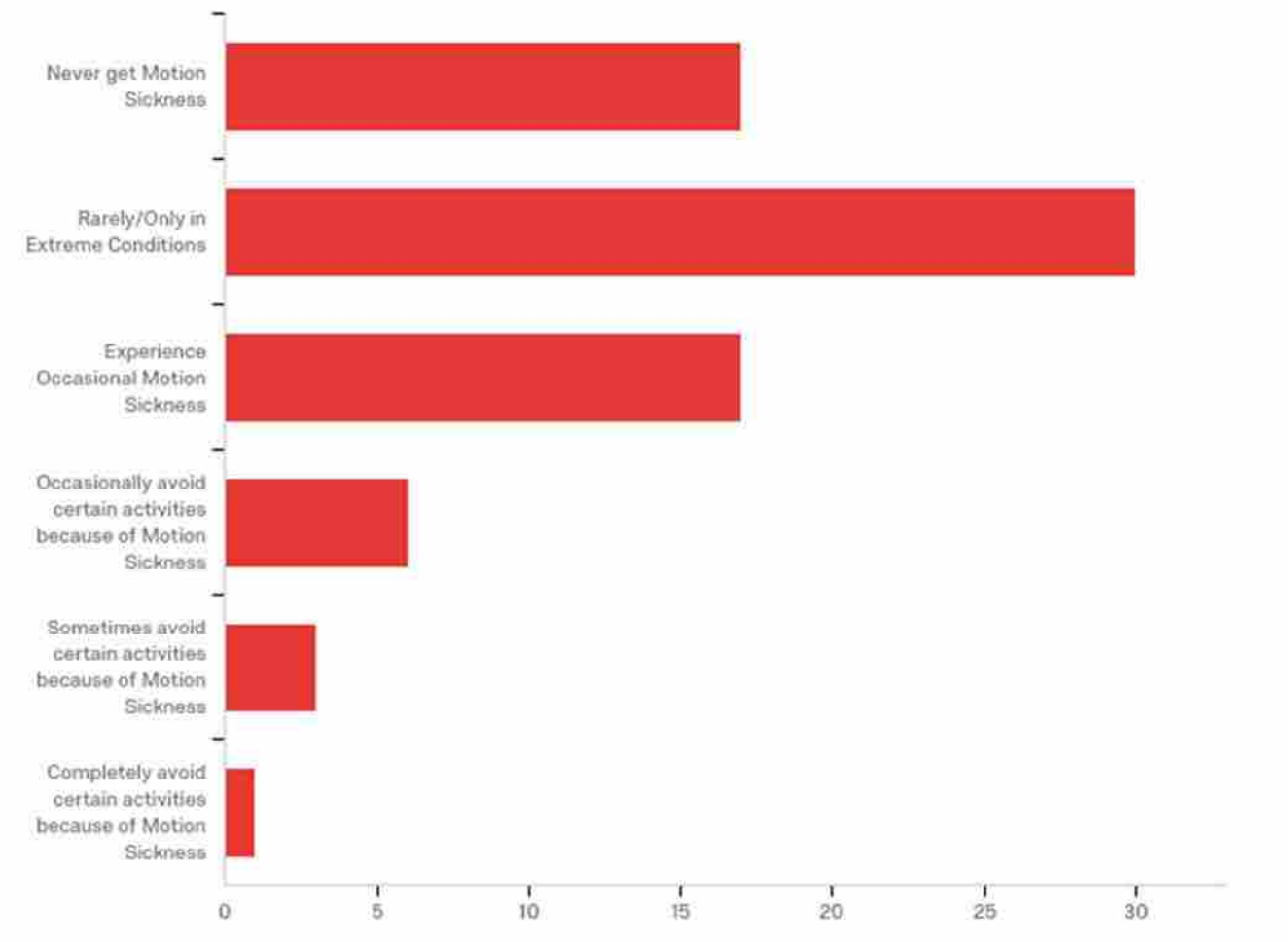
Q11 - Please list how much experience you have with the following items in your lifetime:



#	Question	0-5		5-10		10-30		30-100		>100		Total
1	Hour of Virtual Reality	78.38%	58	13.51%	10	1.35%	1	5.41%	4	1.35%	1	74
2	Hours playing Video games	2.70%	2	1.35%	1	12.16%	9	14.86%	11	68.92%	51	74

3	Hours watching 3D Movies/TV	28.38%	21	33.78%	25	24.32%	18	8.11%	6	5.41%	4	74
4	Hours watching Non-3D Movies/TV	0.00%	0	1.35%	1	0.00%	0	6.76%	5	91.89%	68	74
5	Hours riding in a car	0.00%	0	0.00%	0	0.00%	0	2.70%	2	97.30%	72	74
6	Rides on a roller-coaster	13.51%	10	22.97%	17	21.62%	16	29.73%	22	12.16%	9	74
7	Rides on a swinging boat style amusement ride	40.54%	30	25.68%	19	21.62%	16	9.46%	7	2.70%	2	74
8	Rides on a spinning style amusement ride	21.62%	16	33.78%	25	24.32%	18	12.16%	9	8.11%	6	74

Q14 - In general, how prone to Motion Sickness would you rate yourself?



#	Answer	%	Count
1	Never get Motion Sickness	22.97%	17
2	Rarely/Only in Extreme Conditions	40.54%	30
3	Experience Occasional Motion Sickness	22.97%	17
4	Occasionally avoid certain activities because of Motion Sickness	8.11%	6
5	Sometimes avoid certain activities because of Motion Sickness	4.05%	3
6	Completely avoid certain activities because of Motion Sickness	1.35%	1
	Total	100%	74

Q13 - Do you have any other feedback you would like to share?

Do you have any other feedback you would like to share?
I found it quite easy to learn how to act and communicate in VR, but it was difficult to master.
It was awesome!
I enjoyed this research study a lot actually! It would have been cool to see some pre-made designs too to see a practical application of this in real life. Or even have color changing capacities.
Very fun. I am impressed how far VR and hand tracking has come.
It was a very enjoyable experience. I think that it has a lot of potential to better engineering.
No
This was fun! Thank you!
None, good job.
None so far!
I am pretty uncomfortable with virtual reality, but this wasn't too bad.
Thanks!
No it was good fun
nope! just that I think it is an amazing idea that will greatly help communication across long distances where real life interaction just isn't possible!
Cool stuff!
I love the VR environment where I can create whatever I want. I want to win the free headsets!
N/A
This was a great experience in general, I look forward to advancements in the field of virtual reality.
It was great
Nope!
I extremely enjoyed the experience thanks
I wish the hand movements were more accurate for starting and stopping drawing.
This was super cool! I really like the 3D environment and it felt very natural
I think that the research you're doing is great and I can't wait to see what you can do with it in the future!

It was really fun and cool to see how VR can be used to create some amazing projects.
Not at this time
This VR crap is cool.
You know that we totally were the best and deserve the VR headsets.
Nope
It was a good experience
I really enjoyed the experiment.
VR is really fun.
It was impressive and would be a great tool for engineering and design.
Loved the test. Excited to see this help while explaining visual items.
nah
Fun experiment, the virtual reality felt pretty natural and was fun to explain in.
Loved the experience.
N/A
nope
This was a fun test.

APPENDIX E. SOURCE CODE FOR PROTOTYPE COLLABORATIVE VIRTUAL ENVIRONMENT

E.1 CameraOrbit.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRStandardAssets.Utils;

namespace VRStandardAssets.Maze
{
    // In the maze scene, the camera rotates around
    // the maze in response to the user swiping. This
    // class handles how the camera moves in response
    // to the swipe. This script is placed on a parent
    // of the camera such that the camera pivots around
    // as this gameobject is rotated.
    public class CameraOrbit : MonoBehaviour
    {
        // This enum represents the way in which the camera will rotate around the maze.
        public enum OrbitStyle
        {
            Smooth, Step, StepWithFade, Manual
        }


        [SerializeField] private OrbitStyle m_OrbitStyle;
        [SerializeField] private float m_RotationIncrement = 45f;           // The amount the
            ↪ camera rotates in response to a swipe.
        [SerializeField] private float m_RotationFadeDuration = 0.2f;       // If fading is
            ↪ enabled, this is the duration of the the fade.
        [SerializeField] private VRCameraFade m_CameraFade;                 // Optional reference
            ↪ to the camera fade script, only required if fading is enabled.
        [SerializeField] private VRInput m_VrInput;                         // Reference to the
            ↪ VRInput to subscribe to swipe events.
        [SerializeField] private MazeGameController m_MazeGameController;  // Reference to the
            ↪ game controller so swiping will only be handled while the game is playing.
```

```
[SerializeField] private Rigidbody m_Rigidbody;                    // Reference to the
    ↪ camera's rigidbody.
[SerializeField] private Camera m_Camera;                          // Reference to the
    ↪ actual camera.
[SerializeField] private float m_SphereRadius = 2f;                // How fast to move
    ↪ during smooth style movements.
[SerializeField] private float m_MovementTime = 3f;                // How long a Smooth/
    ↪ Pathed transition should take.
[SerializeField] private Material m_FadeMaterial;                  // The material to
    ↪ use for highlighting the part in question
[SerializeField] private float m_FadeDuration = 1f;                // How long the
    ↪ material should take to fade
[SerializeField] private float m_MovementSpeed = .01f;             // Mupltiplier for
    ↪ joystick movements when the player hasd control.
[SerializeField] private GameObject m_Cone;                        // Mupltiplier for
    ↪ joystick movements when the player hasd control.
[SerializeField] private GameObject m_Open;                        // The Model of the
    ↪ car with everything Open
[SerializeField] private GameObject m_Closed;                      // The Model of the
    ↪ car with everything Closed

private Quaternion m_StartRotation;                                // The rotation of
    ↪ the camera at the start of the scene, used for reseting.

private bool moving = false;
[SerializeField] private bool manualMove = true;
private Queue<Movement> m_MovementList;
private List<Material> m_OriginalMaterials;                        //The original
    ↪ materials for the parts we should be fading in and out.
private List<Renderer> m_FadeRenderers;                            //The renderers for
    ↪ the meshes we should be fading in and out.
private List<GameObject> m_Deactivated;

private int m_MoveNum = 0;
private float m_MovementStartTime = 0f;
private string m_TimeFileRecord = "";

private void Awake ()
{
    // Store the start rotation.
    m_StartRotation = m_Rigidbody.rotation;
    m_MovementList = new Queue<Movement>();
    m_OriginalMaterials = new List<Material>();
    m_FadeRenderers = new List<Renderer>();
    m_Deactivated = new List<GameObject>();
```

```
    //delete the duration file if it exists
    System.IO.File.Delete("C:\\Users\\VR\\Documents\\MovementDuration.csv");
}

private void OnEnable ()
{
    m_VrInput.OnSwipe += HandleSwipe;
    //m_VrInput.OnClick += HandleClick;

}

private void OnDisable ()
{
    m_VrInput.OnSwipe -= HandleSwipe;
    //m_VrInput.OnClick -= HandleClick;
}

private void Update()
{
    //fade the materials between their orginal and the fade material
    float lerp = Mathf.PingPong(Time.time, m_FadeDuration) / m_FadeDuration;
    for (int i = 0; i < m_FadeRenderers.Count; i++)
        m_FadeRenderers[i].material.Lerp(m_OriginalMaterials[i], m_FadeMaterial, lerp);

    if (manualMove) Move();

    if (Input.GetKeyDown("space"))
    {
        //log Timer
        float time = Time.time - m_MovementStartTime;
        string entry = m_MoveNum + "," + time + Environment.NewLine;
        //m_TimeFileRecord += entry;
        Debug.Log("Movement #:" + m_MoveNum + "Time Since Movement Started:" + time);
        System.IO.File.AppendAllText("C:\\Users\\VR\\Documents\\MovementDuration.csv",
            ↪ entry);
    }

    if (Input.GetKeyDown("enter"))
    {
        HandleClick();
    }
}

private void Move()
```

```
{

    if (true)
    {
        //Style 1
        float forward = Input.GetAxis("Forward");
        float right = Input.GetAxis("Right");
        float up = Input.GetAxis("Up");
        float down = Input.GetAxis("Down");
        float rotate = Input.GetAxis("Rotate");
        float rotateUp = Input.GetAxis("RotateUp");

        //up and down are triggers so process here
        up = (up + 1) / 2;
        down = (down + 1) / 2;

        transform.position += m_Camera.transform.forward.normalized * forward *
            ↪ m_MovementSpeed;
        transform.position += m_Camera.transform.right.normalized * right *
            ↪ m_MovementSpeed;
        transform.position += m_Camera.transform.up.normalized * (up - down) *
            ↪ m_MovementSpeed;
        transform.RotateAround(transform.position, m_Camera.transform.up, rotate);
        transform.RotateAround(transform.position, m_Camera.transform.right, rotateUp);
    }
    else
    {
        //Style 2
        float forward = Input.GetAxis("Forward");
        float right = Input.GetAxis("Right");
        float up = Input.GetAxis("RotateUp");
        float rotate = Input.GetAxis("Rotate");

        transform.position += m_Camera.transform.forward.normalized * forward *
            ↪ m_MovementSpeed;
        transform.position += m_Camera.transform.right.normalized * right *
            ↪ m_MovementSpeed;
        transform.position += m_Camera.transform.up.normalized * (up) * m_MovementSpeed;
        transform.RotateAround(transform.position, m_Camera.transform.up, rotate);
    }
}

private void HandleSwipe(VRInput.SwipeDirection swipeDirection)
{
```

```
        // If the game isn't playing or the camera is fading, return and don't handle the
            ↪ swipe.
        if (!m_MazeGameController.Playing)
            return;

        if (m_CameraFade.IsFading)
            return;

        // Otherwise start rotating the camera with either a positive or negative increment.
        switch (swipeDirection)
        {
            case VRInput.SwipeDirection.LEFT:
                StartCoroutine(RotateCamera(m_RotationIncrement));
                break;

            case VRInput.SwipeDirection.RIGHT:
                StartCoroutine(RotateCamera(-m_RotationIncrement));
                break;
        }
    }


    private IEnumerator RotateCamera(float increment)
    {
        // Determine how the camera should rotate base on it's orbit style.
        switch (m_OrbitStyle)
        {
            // If the style is smooth add a torque to the camera's rigidbody.
            case OrbitStyle.Smooth:
                m_Rigidbody.AddTorque (transform.up * increment);
                break;

            // If the style is step then rotate the camera's transform by a set amount.
            case OrbitStyle.Step:
                transform.Rotate(0, increment, 0);
                break;

            // If the style is step with a fade, wait for the camera to fade out, then step
                ↪ the rotation around, the wait for the camera to fade in.
            case OrbitStyle.StepWithFade:
                yield return StartCoroutine(m_CameraFade.BeginFadeOut(m_RotationFadeDuration,
                    ↪  false));
                transform.Rotate(0, increment, 0);
                yield return StartCoroutine(m_CameraFade.BeginFadeIn(m_RotationFadeDuration,
                    ↪ false));
```

```
            break;
        }
    }

    private void HandleClick()
    {
        // If the game isn't playing or the camera is fading, return and don't handle the
            ↪ swipe.
        if (!m_MazeGameController.Playing)
            return;

        if (m_CameraFade.IsFading)
            return;

        if (moving)
            return;

        //If the movement list is empty end the simulation.
        if (m_MovementList.Count == 0)
        {
            //TODO: end the simulation
            return;
        }

        // Otherwise start rotating the camera with either a positive or negative increment.
        Movement move = m_MovementList.Dequeue();
        m_MovementList.Enqueue(move); //infinite loop for testing
        StartCoroutine(MoveCamera(move));
    }

    private IEnumerator MoveCamera (Movement move)
    {
        moving = true;
        //reactivate any components that were deactivated in the last move
        foreach (GameObject obj in m_Deactivated)
            obj.SetActive(true);
        m_Deactivated.Clear();

        // Switch the model of the car that is active
        m_Open.SetActive(move.open);
        m_Closed.SetActive(!move.open);

        //Move the cone
        m_Cone.transform.position = move.coneTipPoint;
        m_Cone.transform.LookAt(move.coneLookPoint);
```

```
//Deactivate the new meshes
foreach(string mesh in move.meshesToHide)
{
    GameObject obj = GameObject.Find(mesh);
    obj.SetActive(false);
    m_Deactivated.Add(obj);
}


// Adjust the Mesh Highlighting
for (int i = 0; i< m_FadeRenderers.Count; i++)
{
    m_FadeRenderers[i].material = m_OriginalMaterials[i];
}
m_FadeRenderers.Clear();
m_OriginalMaterials.Clear();
foreach (string mesh in move.meshesToFade)
{
    Renderer rend = GameObject.Find(mesh).GetComponent<Renderer>();
    if (rend != null)
    {
        m_OriginalMaterials.Add(rend.material);
        rend.material = new Material(rend.material);
        m_FadeRenderers.Add(rend);
    }
}

//if the previous move style was manual and the next one isn't, turn off manual move
    ↪ and reorient the camera to the parent
if( manualMove && move.style != OrbitStyle.Manual)
{
    manualMove = false;
    //m_Cone.SetActive(false);
    //transform.position = m_Camera.transform.position;
    //transform.rotation = m_Camera.transform.rotation;
    //UnityEngine.VR.InputTracking.Recenter();
}

//record the movement start time and increment the movement number
m_MoveNum++;
m_MovementStartTime = Time.time;


// Determine how the camera should rotate base on it's orbit style.
```

```
        switch (move.style)
        {
            case OrbitStyle.Manual:
                manualMove = true;
                //activate and orient the cone to point at the blinking part from the
                    ↪ viewpoint
                //m_Cone.SetActive(true);
                //Vector3 lookDir = move.endLookPoint - move.endLocation;
                //m_Cone.transform.position = move.endLookPoint;
                //m_Cone.transform.rotation = Quaternion.LookRotation(lookDir);
                break;

            // If the style is smooth Lerp along the path
            case OrbitStyle.Smooth:
                yield return StartCoroutine(SmoothMove2(CalcPath3(move), move));
                break;

            // If the style is step then instantly move the camera to the correct location
                ↪ and look point.
            case OrbitStyle.Step:
                transform.position = move.endLocation;
                transform.LookAt(move.endLookPoint);
                break;

            // If the style is step with a fade, wait for the camera to fade out, then move
                ↪ the camera, the wait for the camera to fade in.
            case OrbitStyle.StepWithFade:
                yield return StartCoroutine(m_CameraFade.BeginFadeOut(m_RotationFadeDuration,
                    ↪  false));
                transform.position = move.endLocation;
                transform.LookAt(move.endLookPoint);
                yield return StartCoroutine(m_CameraFade.BeginFadeIn(m_RotationFadeDuration,
                    ↪ false));
                break;
        }
        moving = false;
    }

    private IEnumerator SmoothMove(List<Vector3> path, Movement move)
    {
        float t = 0.0f;
        float time = 0f;

        //Get the camera rotation for various points along the to interpolate between
        //Transform initialTransform = transform;
```

```
        Quaternion startRotation = transform.rotation;
        transform.position = path[2];
        transform.LookAt(Vector3.zero);
        Quaternion intermediateRotation = transform.rotation;
        transform.position = path[4];
        transform.LookAt(move.endLookPoint);
        Quaternion finalRotation = transform.rotation;

        while (time < m_MovementTime)
        {
            time += Time.deltaTime;
            t = Mathf.SmoothStep(0, 2, time / m_MovementTime);
            Vector3 newPos;
            Quaternion newRotation;
            if (t < 1f)
            {
                newPos = (1f - t) * (1f - t) * path[0] + 2 * t * (1f - t) * path[1] + t * t *
                    ↪ path[2];
                newRotation = Quaternion.Lerp(startRotation, intermediateRotation, t);
            }
            else
            {
                float t1 = t - 1f;
                newPos = (1f - t1) * (1f - t1) * path[2] + 2* t1 * (1f - t1) * path[3] + t1 *
                    ↪ t1 * path[4];
                newRotation = Quaternion.Lerp(intermediateRotation, finalRotation, t1);
            }
            transform.position = newPos;
            transform.rotation = newRotation;
            yield return null;
        }
        transform.position = move.endLocation;
        transform.LookAt(move.endLookPoint);
    }

    public void Restart ()
    {
        // To restart, make sure the rotation is reset and the camera is not moving or
            ↪ rotating.
        m_Rigidbody.rotation = m_StartRotation;
        m_Rigidbody.angularVelocity = Vector3.zero;
        m_Rigidbody.velocity = Vector3.zero;
        m_MovementList.Clear();
        ReadCameraFile();
    }
```

```
private void ReadCameraFile()
{
    System.IO.StreamReader input = new System.IO.StreamReader("c:\\Locations.txt");
    string line;
    while ((line = input.ReadLine()) != null)
        m_MovementList.Enqueue(new Movement(line));
}

private List<Vector3> CalcPath(Movement end)
{
    Vector3 curPoint = transform.position;
    Vector3 curDirection = -transform.forward;
    Vector3 endPoint = end.endLocation;
    Vector3 endDirection = end.endLocation - end.endLookPoint;
    Vector3 point1 = Vector3.zero;
    Vector3 point2 = Vector3.zero;
    Vector3 point3 = Vector3.zero;
    List<Vector3> path;
    //if our direction vectors are close to opposite, just fly over the top.
    if (Vector3.Angle(curDirection,endDirection) > 178f)
    {
        point2 = Vector3.up.normalized * m_SphereRadius;
        point1 = curPoint + curDirection.normalized * m_SphereRadius / 2;
        point3 = endPoint + endDirection.normalized * m_SphereRadius / 2;
        path = new List<Vector3> { curPoint, point1, point2, point3, endPoint };
        return path;
    }

    float distanceAlongVector = .0625f;
    float distanceToOrigin = 0.0f;
    //step out along our direction vectors until we find a line that is at least 3m away
        ↪ from the center
    //This may not be necessary...
    do
    {
        distanceAlongVector *= 2;
        float t1 = distanceAlongVector / curDirection.magnitude;
        float t3 = distanceAlongVector / endDirection.magnitude;
        point1 = curPoint + t1 * curDirection;
        point3 = endPoint + t3 * endDirection;

        distanceToOrigin = Vector3.Magnitude(Vector3.Cross(-point1, -point3)) / Vector3.
            ↪ Magnitude(point3 - point1);
    } while (distanceToOrigin < m_SphereRadius);
```

```
        float lower = 0f;
        float upper = distanceAlongVector;

        //perform binary search to find a line that is close to the surface of a 3m sphere,
            ↪ parrallel to the sphere, and passes through our start and end vectors
        while (Math.Abs(m_SphereRadius - distanceToOrigin) > 0.01)
        {
            float distance = (lower + upper) / 2;
            point1 = CalcPointAlongLine(curPoint, curDirection, distance);
            point3 = CalcPointAlongLine(endPoint, endDirection, distance);

            distanceToOrigin = CalcLineToOriginDistance(point1, point3);
            if (distanceToOrigin < m_SphereRadius)
                lower = distance;
            else
                upper = distance;
        }

        //now that we have p1 and p3 find where it touchs the sphere (p2)
        float t = -Vector3.Dot(point1, point3 - point1) / (float) Math.Pow(Vector3.Magnitude
            ↪ (point3 - point1), 2f);
        point2 = point1 + t * (point3 - point1);

        path = new List<Vector3> { curPoint, point1, point2, point3, endPoint };
        return path;
    }

    private float CalcLineToOriginDistance(Vector3 p1, Vector3 p2)
    {
        return Vector3.Magnitude(Vector3.Cross(-p1, -p2)) / Vector3.Magnitude(p2 - p1);
    }

    private Vector3 CalcPointAlongLine(Vector3 start, Vector3 direction, float distance)
    {
        float t = distance / direction.magnitude;
        return start + t * direction;
    }

    private List<Vector3> CalcPath2(Movement end)
    {
        //find the points (a,b) where the current location and the ending location intersect
            ↪ the sphere
        Vector3 a = CalcLineSphereIntersection(transform.position, transform.forward,
            ↪ m_SphereRadius);
```

```
        Vector3 b = CalcLineSphereIntersection(end.endLocation, end.endLookPoint - end.
            ↪ endLocation, m_SphereRadius);
        Vector3 c = Vector3.zero; //the path midpoint
        Vector3 d = Vector3.zero; //the 2nd path point
        Vector3 e = Vector3.zero; //the 4th path point
        double distanceDiff = 0;
        int counter = 0;
        do
        {
            //find the halfway point (c) of a & b on the sphere
            c = (a + b).normalized * m_SphereRadius;
            if (c == Vector3.zero) c = Vector3.up.normalized * m_SphereRadius;
            //c represents the tangent plane point and normal vector
            //find the intersection of the current location and ending location with the
                ↪ plane (d,e)
            try
            {
                d = CalcLinePlaneIntersection(c, c, transform.forward, transform.position);
                e = CalcLinePlaneIntersection(c, c, end.endLookPoint - end.endLocation, end.
                    ↪ endLocation);
            }
            catch (Exception excep)
            {
                Debug.Log(excep.Message);
                d = a;
                e = b;
                break;
            }

            //find the distance of (cd) and (ce)
            float distanceCD = Vector3.Magnitude(c - d);
            float distanceCE = Vector3.Magnitude(c - e);
            //distanceDiff = cd-ce
            distanceDiff = distanceCD - distanceCE;
            if (distanceDiff > 0)
                b = c;
            else
                a = c;
            counter++;
        }
        while (Math.Abs(distanceDiff) > 1e-3 && counter < 100);
        if (counter == 100) Debug.Log("Couldn't find a center point.");
        return new List<Vector3> { transform.position, d, c, e, end.endLocation };
    }
```

```
private Vector3 CalcLinePlaneIntersection(Vector3 planeNormal, Vector3 planePoint,
    ↪ Vector3 lineDir, Vector3 linePoint)
{
    float denom = Vector3.Dot(planeNormal.normalized, lineDir.normalized);
    if (Math.Abs(denom) < 1e-6)
        throw new Exception("Line and Plane are parallel");

    Vector3 linePlaneOrigin = planePoint - linePoint;
    float t = Vector3.Dot(linePlaneOrigin, planeNormal.normalized) / denom;
    Vector3 intersection = linePoint + t * lineDir.normalized;
    return intersection;
}
private Vector3 CalcLineSphereIntersection(Vector3 lineStart, Vector3 lineDir, float
    ↪ sphereRadius)
{
    double a = Vector3.Dot(lineDir.normalized, lineDir.normalized);
    double b = 2 * Vector3.Dot(lineDir.normalized, lineStart);
    double c = Vector3.Dot(lineStart, lineStart) - sphereRadius * sphereRadius;

    double distance = 0;

    try
    {
        distance = solveQuadratic(a, b, c);
    }
    catch (Exception e)
    {
        Debug.Log(e.Message);
        return Vector3.zero;
    }

    return lineStart + (float)distance * lineDir.normalized;
}
private double solveQuadratic(double a, double b, double c)
{
    double discriminant = b * b - 4 * a * c;
    if (discriminant < 0.0)
        throw new Exception("Imaginary intersection");
    else
    {
        double q = b > 0.0 ?
            -0.5 * (b + Math.Sqrt(discriminant)) :
            -0.5 * (b - Math.Sqrt(discriminant));

        double x0 = q / a;
```

```
            double x1 = c / q;

            if (x0 * x1 > 0) //both intersections have the same sign
                throw new Exception("Ray issue");
            else if (x0 < 0)
                return x0;
            else
                return x1;
        }
    }

    private IEnumerator SmoothMove2(List<Vector3> path, Movement move)
    {
        //First calc the total path length
        float[] bez1LengthMap = approxBezierPathLength(path[0], path[1], path[2]);
        float[] bez2LengthMap = approxBezierPathLength(path[3], path[4], path[5]);
        float bez1Length = bez1LengthMap[bez1LengthMap.Length - 1];
        float bez2Length = bez2LengthMap[bez2LengthMap.Length - 1];
        float arcLength = m_SphereRadius * Vector3.Angle(path[2], path[3]) * (float)Math.PI /
            ↪ 180f;
        float totalLength = bez1Length + bez2Length + arcLength;

        //now find any needed intermediate rotations
        Quaternion start = transform.rotation;
        transform.position = move.endLocation;
        transform.LookAt(move.endLookPoint);
        Quaternion end = transform.rotation;

        float t1 = 0f,t2 = 0f;
        float time = 0f;
        Vector3 newPos;

        while (time < m_MovementTime)
        {
            time += Time.deltaTime;
            if (time > m_MovementTime) time = m_MovementTime;
            float dist = Mathf.SmoothStep(0, totalLength, time / m_MovementTime);

            if (dist <= bez1Length)
            {
                //calc new t to get a constant movement speed
                t1 = findTValueForDistance(dist, bez1LengthMap);
                newPos = calcBezierPathPoint(t1 , path[0], path[1], path[2]);
            }
            else if (dist <= bez1Length + arcLength)
```

```
            {
                newPos = Vector3.Slerp(path[2], path[3], (dist - bez1Length) / arcLength);
            }
            else
            {
                //calc new t to get a constant movement speed
                t2 = findTValueForDistance(dist-bez1Length-arcLength, bez2LengthMap);
                newPos = calcBezierPathPoint(t2, path[3], path[4], path[5]);
            }

            transform.position = newPos;
            transform.rotation = Quaternion.Slerp(start, end, dist / totalLength);
            yield return null;
        }
        transform.position = move.endLocation;
        transform.LookAt(move.endLookPoint);
    }

    private List<Vector3> CalcPath3(Movement end)
    {
        const float d = 10f;//meters

        //First find where our current point/view intersects the sphere
        Vector3 a = CalcLineSphereIntersection(transform.position, transform.forward,
            ↪ m_SphereRadius);
        Vector3 b = CalcLineSphereIntersection(end.endLocation, end.endLookPoint - end.
            ↪ endLocation, m_SphereRadius);

        //now find the arc length
        float arcLength = m_SphereRadius * Vector3.Angle(a, b) * (float)Math.PI / 180f;

        Vector3 point3;
        Vector3 point4;
        //now find the second endpoint of each bezier curve
        //note that is arcLength isn't greater than 2*d we split the difference to avoid a
            ↪ wierd bouncing path
        if (arcLength >= d*2)
        {
            point3 = Vector3.Slerp(a, b, d / arcLength);
            point4 = Vector3.Slerp(b, a, d / arcLength);
        }
        else
        {
            point3 = point4 = Vector3.Slerp(a, b, 0.5f);
        }
```

```
        //Next find the middle point of each path
        Vector3 point2 = CalcLinePlaneIntersection(point3, point3, transform.forward,
            ↪ transform.position);
        Vector3 point5 = CalcLinePlaneIntersection(point4, point4, end.endLookPoint - end.
            ↪ endLocation, end.endLocation);

        List<Vector3> path = new List<Vector3> { transform.position, point2, point3, point4,
            ↪ point5, end.endLocation };
        return path;
    }

    private float findTValueForDistance(float distance, float[] values)
    {
        int low = 0;
        int high = values.Length - 1;

        if (distance <= 0) return 0f;
        if (distance >= values[high]) return 1f;

        while (low < high)
        {
            int index = (low + high) / 2;

            //binary search to value
            if (distance < values[index])
                high = index;
            else if (distance > values[index])
                low = index;
            else
                return ((float)index) / ((float)values.Length);

            //if high == low + 1 interpolate to value
            if(high == low +1)
            {
                float value = Mathf.Lerp(low, high, (distance - values[low]) / (values[high]
                    ↪ - values[low]))/values.Length;

                if (value < 0)
                {
                    Debug.LogError("t<0");
                    value = 0f;
                }
                if (value > 1)
                {
```

```
                Debug.LogError("t>1");
                value = 1f;
            }

            return value;
        }
    }
    return -1;
}

private float[] approxBezierPathLength(Vector3 start, Vector3 middle, Vector3 end)
{
    float length = 0;
    Vector3 prevPoint = Vector3.zero;
    Vector3 currentPoint = Vector3.zero;

    const int numSegments = 10000;
    float[] result = new float[numSegments + 1];

    for (int i = 0; i <= numSegments; i++)
    {
        currentPoint = calcBezierPathPoint((float)i/numSegments, start, middle, end);

        if(i>0) length += Vector3.Magnitude(currentPoint-prevPoint);

        result[i] = length;
        prevPoint = currentPoint;
    }
    return result;
}

private Vector3 calcBezierPathPoint(float t, Vector3 start, Vector3 middle, Vector3 end)
{
    return (1f - t) * (1f - t) * start
        + 2f * t * (1f - t) * middle
        + t * t * end;
}


private class Movement
{
    public OrbitStyle style;
    public Vector3 endLocation;
    public Vector3 endLookPoint;
    public Vector3 coneTipPoint;
```

```
public Vector3 coneLookPoint;
public bool open;
public List<string> meshesToFade;
public List<string> meshesToHide;

public Movement (string move)
{
    char[] delims = new char[] { ' ', '\t', '\r', '\n' };
    string[] results = move.Split(delims, System.StringSplitOptions.
        ↪ RemoveEmptyEntries);

    SetStyle(results[0]);
    endLocation = SetPoint(results[1], results[2], results[3]);
    endLookPoint = SetPoint(results[4], results[5], results[6]);
    coneTipPoint = SetPoint(results[7], results[8], results[9]);
    coneLookPoint = SetPoint(results[10], results[11], results[12]);
    open = results[13].ToLower() == "open";
    meshesToFade = new List<string>();
    int i = 14;
    for (; i<results.Length; i++)
    {
        if (results[i] == "&")
            break;
        meshesToFade.Add(results[i]);
    }
    i++;
    meshesToHide = new List<string>();
    for (;i<results.Length; i++)
    {
        meshesToHide.Add(results[i]);
    }

}
private Vector3 SetPoint(string v1, string v2, string v3)
{
    float x = float.Parse(v1);
    float y = float.Parse(v2);
    float z = float.Parse(v3);

    return new Vector3(x, y, z);
}

private void SetStyle(string ss)
{
    switch (ss)
```

```
                {
                    case "j":
                        style = OrbitStyle.Step;
                        break;
                    case "f":
                        style = OrbitStyle.StepWithFade;
                        break;
                    case "p":
                        style = OrbitStyle.Smooth;
                        break;
                    case "u":
                        style = OrbitStyle.Manual;
                        break;
                    default:
                        break;

                }
            }
        }
    }
}
```

APPENDIX F. SOURCE CODE FOR PROTOTYPE COLLABORATIVE VIRTUAL ENVIRONMENT

F.1 ButtonHandler.cs

```
using UnityEngine;
using System.Collections.Generic;
using Leap.Unity.DetectionExamples;

public class ButtonHandler : MonoBehaviour
{

    public void StartRecording()
    {
        foreach (GameObject player in GameObject.FindGameObjectsWithTag("Player"))
        {
            NetworkHandController controller = player.GetComponent<NetworkHandController>();
            if (controller != null)
                controller.RaiseCurtain();

            PinchDraw draw = player.GetComponent<PinchDraw>();
            if (draw != null)
                draw.BreakConnection();

            Deletor deletor = player.GetComponent<Deletor>();
            if (deletor != null)
                deletor.enabled = false;

            Tracker tracker = player.GetComponent<Tracker>();
            if (tracker != null)
                tracker.SetLog(true);
        }

    }

    public void StopRecording()
    {
        foreach (GameObject player in GameObject.FindGameObjectsWithTag("Player"))
        {
```

```
        NetworkHandController controller = player.GetComponent<NetworkHandController>();
        if (controller != null)
            controller.LowerCurtain();

        PinchDraw draw = player.GetComponent<PinchDraw>();
        if (draw != null)
            draw.MakeConnection();

        Deletor deletor = player.GetComponent<Deletor>();
        if (deletor != null)
            deletor.enabled = true;

        Tracker tracker = player.GetComponent<Tracker>();
        if (tracker != null)
            tracker.SetLog(false);

        Transform lines = GameObject.Find("Lines").transform;
        foreach (Transform child in lines.transform)
        {
            Destroy(child.gameObject);
        }
    }
}

private GameObject tutorial = null;

public void hideTutorial()
{
    if (tutorial == null)
        tutorial = GameObject.Find("Tutorial");
    if (tutorial != null)
        tutorial.SetActive(false);
    foreach (NetworkHandController cont in GameObject.FindObjectsOfType<NetworkHandController
        ↪ >())
        cont.RpcHideTutorial();
}

public void showTutorial()
{
    if (tutorial == null)
        tutorial = GameObject.Find("Tutorial");
    if (tutorial != null)
        tutorial.SetActive(true);
    foreach (NetworkHandController cont in GameObject.FindObjectsOfType<NetworkHandController
        ↪ >())
```

```
            cont.RpcShowTutorial();
    }
}
```

F.2 Deletor.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.Networking;

public class Deletor : NetworkBehaviour {

        [SerializeField]
        private Tracker tracker;

        [Command]
        public void CmdDelete (string name)
        {
                tracker.LogDelete(name);
                Destroy(GameObject.Find(name));
        if (!enabled) return;
        RpcDelete(name);
        }

        [ClientRpc]
        void RpcDelete(string name)
        {
                if (isLocalPlayer) return;
                Destroy(GameObject.Find(name));
        }
}
```

F.3 EventCapsuleHand.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using Leap;
using System;

namespace Leap.Unity
{
```

```
/** A basic Leap hand model constructed dynamically vs. using pre-existing geometry */

public delegate void HandPositionChanged(object sender, List<Vector3> positions);
public delegate void GeneratedMeshes(object sender, Chirality hand);

public class EventCapsuleHand : IHandModel
{

    private const int THUMB_BASE_INDEX = (int)Finger.FingerType.TYPE_THUMB * 4;
    private const int PINKY_BASE_INDEX = (int)Finger.FingerType.TYPE_PINKY * 4;

    private const float SPHERE_RADIUS = 0.008f;
    private const float CYLINDER_RADIUS = 0.006f;
    private const float PALM_RADIUS = 0.015f;

    [SerializeField]
    private Chirality handedness;

    [SerializeField]
    private bool _showArm = true;

    [SerializeField]
    private Material _material;

    [SerializeField]
    private Mesh _sphereMesh;

    [SerializeField]
    private int _cylinderResolution = 12;

    public Color HandColor
    {
        get { return jointMat.color; }
        set { jointMat.color = value; }
    }
    private bool _hasGeneratedMeshes = false;
    private Material jointMat;

    [SerializeField, HideInInspector]
    private List<Transform> _serializedTransforms;

    private Transform[] _jointSpheres;
    private Transform mockThumbJointSphere;
    private Transform palmPositionSphere;
```

```
    private Transform wristPositionSphere;

    private List<Renderer> _armRenderers;
    private List<Transform> _cylinderTransforms;
    private List<Transform> _sphereATransforms;
    private List<Transform> _sphereBTransforms;

    private Transform armFrontLeft, armFrontRight, armBackLeft, armBackRight;
    private Hand hand_;

    public event HandPositionChanged handPositionChanged;
    public event GeneratedMeshes generatedMeshes;

    void OnHandChanged(List<Vector3> list)
    {
        if(handPositionChanged != null)
        {
            handPositionChanged(this, list);
        }
    }

    void OnGeneratedMeshes()
    {
        if (generatedMeshes != null)
            generatedMeshes(this, handedness);
    }

    public void FixedUpdate()
    {
        if (_jointSpheres == null) return;
        List<Vector3> positions = new List<Vector3>();
        foreach (Transform t in _jointSpheres)
            positions.Add(t.position);
        positions.Add(palmPositionSphere.position);
        positions.Add(wristPositionSphere.position);
        positions.Add(mockThumbJointSphere.position);
        OnHandChanged(positions);
    }

    public override ModelType HandModelType
    {
        get
        {
            return ModelType.Graphics;
        }
```

```
}

public override Chirality Handedness
{
    get
    {
        return handedness;
    }
    set { }
}

public override bool SupportsEditorPersistence()
{
    return true;
}

public override Hand GetLeapHand()
{
    return hand_;
}

public override void SetLeapHand(Hand hand)
{
    hand_ = hand;
}

void OnValidate()
{
    //Resolution must be at least 3!
    _cylinderResolution = Mathf.Max(3, _cylinderResolution);

    //Update visibility on validate so that the user can toggle in real-time
    if (_armRenderers != null)
    {
        updateArmVisibility();
    }
}

void Awake()
{
    if (_material != null)
    {
        jointMat = new Material(_material);
        jointMat.hideFlags = HideFlags.DontSaveInEditor;
    }
```

```
}

public override void InitHand()
{
    if (_serializedTransforms != null)
    {
        for (int i = 0; i < _serializedTransforms.Count; i++)
        {
            var obj = _serializedTransforms[i];
            if (obj != null)
            {
                DestroyImmediate(obj.gameObject);
            }
        }
        _serializedTransforms.Clear();
    }
    else
    {
        _serializedTransforms = new List<Transform>();
    }

    _jointSpheres = new Transform[4 * 5];
    _armRenderers = new List<Renderer>();
    _cylinderTransforms = new List<Transform>();
    _sphereATransforms = new List<Transform>();
    _sphereBTransforms = new List<Transform>();

    createSpheres();
    createCylinders();

    updateArmVisibility();

    _hasGeneratedMeshes = false;
}

public override void BeginHand()
{
    base.BeginHand();

    //if (hand_.IsLeft)
    //{
    //    jointMat.color = _leftColorList[_leftColorIndex];
    //    _leftColorIndex = (_leftColorIndex + 1) % _leftColorList.Length;
    //}
    //else
```

```
        //{
        //    jointMat.color = _rightColorList[_rightColorIndex];
        //    _rightColorIndex = (_rightColorIndex + 1) % _rightColorList.Length;
        //}
    }

    public override void UpdateHand()
    {
        //Update the spheres first
        updateSpheres();

        //Update Arm only if we need to
        if (_showArm)
        {
            updateArm();
        }

        //The cylinder transforms are deterimined by the spheres they are connected to
        updateCylinders();
    }

    //Transform updating methods

    private void updateSpheres()
    {
        //Update all spheres
        List<Finger> fingers = hand_.Fingers;
        for (int i = 0; i < fingers.Count; i++)
        {
            Finger finger = fingers[i];
            for (int j = 0; j < 4; j++)
            {
                int key = getFingerJointIndex((int)finger.Type, j);
                Transform sphere = _jointSpheres[key];
                sphere.position = finger.Bone((Bone.BoneType)j).NextJoint.ToVector3();
            }
        }

        palmPositionSphere.position = hand_.PalmPosition.ToVector3();

        Vector3 wristPos = hand_.PalmPosition.ToVector3();
        wristPositionSphere.position = wristPos;

        Transform thumbBase = _jointSpheres[THUMB_BASE_INDEX];
```

```
        Vector3 thumbBaseToPalm = thumbBase.position - hand_.PalmPosition.ToVector3();
        mockThumbJointSphere.position = hand_.PalmPosition.ToVector3() + Vector3.Reflect(
            ↪ thumbBaseToPalm, hand_.Basis.xBasis.ToVector3());

        //List<Vector3> positions = new List<Vector3>();
        //foreach (Transform t in _jointSpheres)
        //    positions.Add(t.position);
        //positions.Add(palmPositionSphere.position);
        //positions.Add(wristPositionSphere.position);
        //positions.Add(mockThumbJointSphere.position);
        //OnHandChanged(positions);
    }

    private void updateArm()
    {
        var arm = hand_.Arm;
        Vector3 right = arm.Basis.xBasis.ToVector3() * arm.Width * 0.7f * 0.5f;
        Vector3 wrist = arm.WristPosition.ToVector3();
        Vector3 elbow = arm.ElbowPosition.ToVector3();

        float armLength = Vector3.Distance(wrist, elbow);
        wrist -= arm.Direction.ToVector3() * armLength * 0.05f;

        armFrontRight.position = wrist + right;
        armFrontLeft.position = wrist - right;
        armBackRight.position = elbow + right;
        armBackLeft.position = elbow - right;
    }

    private void updateCylinders()
    {
        for (int i = 0; i < _cylinderTransforms.Count; i++)
        {
            Transform cylinder = _cylinderTransforms[i];
            Transform sphereA = _sphereATransforms[i];
            Transform sphereB = _sphereBTransforms[i];

            Vector3 delta = sphereA.position - sphereB.position;

            if (!_hasGeneratedMeshes)
            {
                MeshFilter filter = cylinder.GetComponent<MeshFilter>();
                filter.sharedMesh = generateCylinderMesh(delta.magnitude / transform.
                    ↪ lossyScale.x);
            }
```

```
            cylinder.position = sphereA.position;

            if (delta.sqrMagnitude <= Mathf.Epsilon)
            {
                //Two spheres are at the same location, no rotation will be found
                continue;
            }

            cylinder.LookAt(sphereB);
        }

        _hasGeneratedMeshes = true;
        OnGeneratedMeshes();
    }

    private void updateArmVisibility()
    {
        for (int i = 0; i < _armRenderers.Count; i++)
        {
            _armRenderers[i].enabled = _showArm;
        }
    }

    //Geometry creation methods

    private void createSpheres()
    {
        //Create spheres for finger joints
        List<Finger> fingers = hand_.Fingers;
        for (int i = 0; i < fingers.Count; i++)
        {
            Finger finger = fingers[i];
            for (int j = 0; j < 4; j++)
            {
                int key = getFingerJointIndex((int)finger.Type, j);
                _jointSpheres[key] = createSphere("Joint", SPHERE_RADIUS);
            }
        }

        mockThumbJointSphere = createSphere("MockJoint", SPHERE_RADIUS);
        palmPositionSphere = createSphere("PalmPosition", PALM_RADIUS);
        wristPositionSphere = createSphere("WristPosition", SPHERE_RADIUS);

        armFrontLeft = createSphere("ArmFrontLeft", SPHERE_RADIUS, true);
```

```
        armFrontRight = createSphere("ArmFrontRight", SPHERE_RADIUS, true);
        armBackLeft = createSphere("ArmBackLeft", SPHERE_RADIUS, true);
        armBackRight = createSphere("ArmBackRight", SPHERE_RADIUS, true);
    }

    private void createCylinders()
    {
        //Create cylinders between finger joints
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                int keyA = getFingerJointIndex(i, j);
                int keyB = getFingerJointIndex(i, j + 1);

                Transform sphereA = _jointSpheres[keyA];
                Transform sphereB = _jointSpheres[keyB];

                createCylinder("Finger Joint", sphereA, sphereB);
            }
        }

        //Create cylinders between finger knuckles
        for (int i = 0; i < 4; i++)
        {
            int keyA = getFingerJointIndex(i, 0);
            int keyB = getFingerJointIndex(i + 1, 0);

            Transform sphereA = _jointSpheres[keyA];
            Transform sphereB = _jointSpheres[keyB];

            createCylinder("Hand Joints", sphereA, sphereB);
        }

        //Create the rest of the hand
        Transform thumbBase = _jointSpheres[THUMB_BASE_INDEX];
        Transform pinkyBase = _jointSpheres[PINKY_BASE_INDEX];
        createCylinder("Hand Bottom", thumbBase, mockThumbJointSphere);
        createCylinder("Hand Side", pinkyBase, mockThumbJointSphere);

        createCylinder("ArmFront", armFrontLeft, armFrontRight, true);
        createCylinder("ArmBack", armBackLeft, armBackRight, true);
        createCylinder("ArmLeft", armFrontLeft, armBackLeft, true);
        createCylinder("ArmRight", armFrontRight, armBackRight, true);
    }
```

```
private int getFingerJointIndex(int fingerIndex, int jointIndex)
{
    return fingerIndex * 4 + jointIndex;
}

private Transform createSphere(string name, float radius, bool isPartOfArm = false)
{
    GameObject sphere = new GameObject(name);
    _serializedTransforms.Add(sphere.transform);

    sphere.AddComponent<MeshFilter>().mesh = _sphereMesh;
    sphere.AddComponent<MeshRenderer>().sharedMaterial = jointMat;
    sphere.transform.parent = transform;
    sphere.transform.localScale = Vector3.one * radius * 2;

    sphere.hideFlags = HideFlags.DontSave | HideFlags.HideInHierarchy | HideFlags.
        ↪ HideInInspector;
    sphere.layer = gameObject.layer;

    if (isPartOfArm)
    {
        _armRenderers.Add(sphere.GetComponent<Renderer>());
    }

    return sphere.transform;
}

private void createCylinder(string name, Transform jointA, Transform jointB, bool
    ↪ isPartOfArm = false)
{
    GameObject cylinder = new GameObject(name);
    _serializedTransforms.Add(cylinder.transform);

    cylinder.AddComponent<MeshFilter>();
    cylinder.AddComponent<MeshRenderer>().sharedMaterial = _material;
    cylinder.transform.parent = transform;

    _cylinderTransforms.Add(cylinder.transform);
    _sphereATransforms.Add(jointA);
    _sphereBTransforms.Add(jointB);

    cylinder.gameObject.layer = gameObject.layer;
    cylinder.hideFlags = HideFlags.DontSave | HideFlags.HideInHierarchy | HideFlags.
        ↪ HideInInspector;
```

```
        if (isPartOfArm)
        {
            _armRenderers.Add(cylinder.GetComponent<Renderer>());
        }
    }

    private Mesh generateCylinderMesh(float length)
    {
        //Debug.Log("================Cyclinder Positions==============");
        // foreach (var sphere in _jointSpheres)
        //     Debug.Log(sphere.position);
        // Debug.Log(palmPositionSphere.position);
        // Debug.Log(wristPositionSphere.position);
        // Debug.Log(mockThumbJointSphere.position);

        //Debug.Log(length);

        Mesh mesh = new Mesh();
        mesh.name = "GeneratedCylinder";
        mesh.hideFlags = HideFlags.DontSave;

        List<Vector3> verts = new List<Vector3>();
        List<Color> colors = new List<Color>();
        List<int> tris = new List<int>();

        Vector3 p0 = Vector3.zero;
        Vector3 p1 = Vector3.forward * length;
        for (int i = 0; i < _cylinderResolution; i++)
        {
            float angle = (Mathf.PI * 2.0f * i) / _cylinderResolution;
            float dx = CYLINDER_RADIUS * Mathf.Cos(angle);
            float dy = CYLINDER_RADIUS * Mathf.Sin(angle);

            Vector3 spoke = new Vector3(dx, dy, 0);

            verts.Add((p0 + spoke) * transform.lossyScale.x);
            verts.Add((p1 + spoke) * transform.lossyScale.x);

            colors.Add(Color.white);
            colors.Add(Color.white);

            int triStart = verts.Count;
            int triCap = _cylinderResolution * 2;
```

```
                tris.Add((triStart + 0) % triCap);
                tris.Add((triStart + 2) % triCap);
                tris.Add((triStart + 1) % triCap);

                tris.Add((triStart + 2) % triCap);
                tris.Add((triStart + 3) % triCap);
                tris.Add((triStart + 1) % triCap);
            }
            mesh.SetVertices(verts);
            mesh.SetIndices(tris.ToArray(), MeshTopology.Triangles, 0);
            mesh.RecalculateBounds();
            mesh.RecalculateNormals();
            mesh.Optimize();
            mesh.UploadMeshData(true);

            return mesh;
        }
    }
}
```

F.4 GeneralClear.cs

```
using UnityEngine;
using System.Collections;

public class GeneralClear : MonoBehaviour {


    private GameObject lines;

    [SerializeField]
    private Deletor deletor;

    // Use this for initialization
    void Start () {
        lines = GameObject.Find("Lines");
    }

    public void ClearLines()
    {
        foreach(Transform line in lines.transform)
        {
            deletor.CmdDelete(line.gameObject.name);
            Destroy(line.gameObject);
```

```
        }
    }
}
```

F.5 HighlightDelete.cs

```
using UnityEngine;
using Leap.Unity;
using UnityEngine.Networking;

public class HighlightDelete : NetworkBehaviour
{

    GameObject _target;
    Material _originalMaterial;

    [SerializeField]
    Material _highlightMaterial;

    [SerializeField]
    Deletor delete;
    public void setTarget(GameObject target)
    {
        if (!enabled) return;
        if (_target == null)
        {
            _target = target;
            _originalMaterial = _target.GetComponent<Renderer>().material;
            _target.GetComponent<Renderer>().material = _highlightMaterial;
        }
    }

    public void pickupTarget()
    {
        if (!enabled) return;
        if (_target == null) return;
        delete.CmdDelete(_target.name);
        Destroy(_target);
        _target = null;
    }


    public void releaseTarget()
    {
```

```
    }

    public void clearTarget()
    {
        if (!enabled) return;
        resetTargetProperties();
        _target = null;
    }

    private void resetTargetProperties()
    {
        if (_target == null) return;
        _target.GetComponent<Renderer>().material = _originalMaterial;
    }

    void OnDisable()
    {
        resetTargetProperties();
        _target = null;

    }
}
```

F.6 LeapDisabler.cs

```
using UnityEngine;
using UnityEngine.Networking;

public class LeapDisabler : NetworkBehaviour {
        [SerializeField] private GameObject leapHandController;
    [SerializeField] private GameObject hands;
    [SerializeField] private Camera mainCamera;
        // Use this for initialization
        void Start () {
        if (!isLocalPlayer)
        {
            leapHandController.SetActive(false);
            hands.SetActive(false);
            mainCamera.gameObject.SetActive(false);
        }
        }
}
```

F.7 NetworkHandController.cs

```
using UnityEngine;
using UnityEngine.Networking;
using System;
using System.Collections;
using System.Collections.Generic;
using Leap;
using Leap.Unity;

public class NetworkHandController : NetworkBehaviour {

    public class SyncListVec3 : SyncListStruct<Vector3> {}

    private const float SPHERE_RADIUS = 0.008f;
    private const float CYLINDER_RADIUS = 0.006f;
    private const float PALM_RADIUS = 0.015f;

    private const int THUMB_BASE_INDEX = (int)Finger.FingerType.TYPE_THUMB * 4;
    private const int PINKY_BASE_INDEX = (int)Finger.FingerType.TYPE_PINKY * 4;

    private Transform[] _jointSpheres_L;
    private Transform[] _jointSpheres_R;
    private Transform mockThumbJointSphere_L;
    private Transform palmPositionSphere_L;
    private Transform wristPositionSphere_L;
    private Transform armFrontLeft_L, armFrontRight_L, armBackLeft_L, armBackRight_L;

    private Transform mockThumbJointSphere_R;
    private Transform palmPositionSphere_R;
    private Transform wristPositionSphere_R;
    private Transform armFrontLeft_R, armFrontRight_R, armBackLeft_R, armBackRight_R;

    [SerializeField, HideInInspector]
    private List<Transform> _serializedTransforms;

    [SerializeField]
    private Mesh _sphereMesh;

    SyncListVec3 syncedPositions_L;
    SyncListVec3 syncedPositions_R;
    [SyncVar]
    bool isEnabled = true;

    [SyncVar]
    bool hasGenMeshesL = false;
```

```
    [SyncVar]
    bool hasGenMeshesR = false;

    [SyncVar]
    public Color color;

    [SyncVar]
    public int PlayerNum = 0;

    [SerializeField]
    private bool _showArm = true;

    [SerializeField]
    protected EventCapsuleHand LeftHandController;

    [SerializeField]
    protected EventCapsuleHand RightHandController;

    [SerializeField]
    private Material _material;

    [SerializeField]
    private int _cylinderResolution = 12;

    private bool _hasGeneratedMeshesL = false;
    private bool _hasGeneratedMeshesR = false;
    private Material jointMat;

    private List<Renderer> _armRenderers;
    private List<Transform> _cylinderTransforms;
    private List<Transform> _sphereATransforms;
    private List<Transform> _sphereBTransforms;

    private Color[] colorList = { Color.blue, Color.green, Color.red, Color.magenta, Color.yellow
        ↪ , Color.gray, Color.cyan, Color.black };

    // Use this for initialization
    void Awake ()
    {
        LeftHandController.handPositionChanged += HandleHandController;
        RightHandController.handPositionChanged += HandleHandController;
        LeftHandController.generatedMeshes += HandleGeneratedMeshes;
        RightHandController.generatedMeshes += HandleGeneratedMeshes;
        syncedPositions_L = new SyncListVec3();
        syncedPositions_R = new SyncListVec3();
```

```
        _jointSpheres_L = new Transform[4 * 5];
        _jointSpheres_R = new Transform[4 * 5];

        PlayerNum = GameObject.FindGameObjectsWithTag("Player").Length;
        color = colorList[PlayerNum % 8];
    }

    void Start () {
        if (isServer)
        {
            for (int i = 0; i < 23; i++)
            {
                syncedPositions_L.Add(Vector3.zero);
                syncedPositions_R.Add(Vector3.zero);
            }
        }
        if (!isLocalPlayer)
        {
            InitHand();
            jointMat.color = color;
        }
        else
        {
            LeftHandController.HandColor = color;
            RightHandController.HandColor = color;
        }

    }

    void OnEnable()
    {
        isEnabled = true;
    }

    void OnDisable()
    {
        isEnabled = false;
    }

    GameObject tutorial = null;

    [ClientRpc]
    public void RpcShowTutorial()
    {
        if (!isLocalPlayer) return;
```

```
    if (tutorial == null)
        tutorial = GameObject.Find("Tutorial");
    if (tutorial != null)
        tutorial.SetActive(true);
}

[ClientRpc]
public void RpcHideTutorial()
{
    if (!isLocalPlayer) return;
    if (tutorial == null)
        tutorial = GameObject.Find("Tutorial");
    if (tutorial != null)
        tutorial.SetActive(false);
}

public void RaiseCurtain()
{
    RpcRaiseCurtain();
}

public void LowerCurtain()
{
    RpcLowerCurtain();
}

[ClientRpc]
void RpcRaiseCurtain()
{
    if (!isLocalPlayer)
    {
        SetRemoteHandsActive(false);
    }
    else
    {
        Transform lines = GameObject.Find("Lines").transform;
        foreach(Transform child in lines.transform)
        {
            Destroy(child.gameObject);
        }
    }
}

[ClientRpc]
void RpcLowerCurtain()
```

```
{
    if (!isLocalPlayer)
    {
        SetRemoteHandsActive(true);
    }
    else
    {
        Transform lines = GameObject.Find("Lines").transform;
        foreach (Transform child in lines.transform)
        {
            Destroy(child.gameObject);
        }
    }
}

void SetRemoteHandsActive(bool active)
{
    foreach(var sphere in _jointSpheres_L)
        sphere.gameObject.SetActive(active);
    foreach (var sphere in _jointSpheres_R)
        sphere.gameObject.SetActive(active);

    mockThumbJointSphere_L.gameObject.SetActive(active);
    palmPositionSphere_L.gameObject.SetActive(active);
    wristPositionSphere_L.gameObject.SetActive(active);

    mockThumbJointSphere_R.gameObject.SetActive(active);
    palmPositionSphere_R.gameObject.SetActive(active);
    wristPositionSphere_R.gameObject.SetActive(active);

    foreach (var bone in _cylinderTransforms)
        bone.gameObject.SetActive(active);
}

// Update is called once per frame
void Update () {
    if(!isLocalPlayer)
    {
        UpdateHand();
    }
}

void HandleGeneratedMeshes(object sender, Chirality hand)
{
    if (hand == Chirality.Left)
```

```
            CmdSetHasGenMeshL(); //StartCoroutine(_WaitForSeconds(0.2f, CmdSetHasGenMeshL));
        else if (hand == Chirality.Right)
            CmdSetHasGenMeshR(); //StartCoroutine(_WaitForSeconds(0.2f, CmdSetHasGenMeshR));
    }

    [Command]
    void CmdSetHasGenMeshL()
    {
        //Debug.Log("CmdSetHasGenMeshL");
        //hasGenMeshesL = true;
        StartCoroutine(SetHasGenMeshLInSecs(.1f));
    }

    [Command]
    void CmdSetHasGenMeshR()
    {
        //Debug.Log("CmdSetHasGenMeshR");
        //hasGenMeshesR = true;
        StartCoroutine(SetHasGenMeshRInSecs(.1f));
    }

    IEnumerator SetHasGenMeshLInSecs(float time)
    {
        yield return new WaitForSecondsRealtime(time);

        hasGenMeshesL = true;
    }

    IEnumerator SetHasGenMeshRInSecs(float time)
    {
        yield return new WaitForSecondsRealtime(time);

        hasGenMeshesR = true;
    }
    void HandleHandController(object sender, List<Vector3> Positions)
    {
        EventCapsuleHand hand = (EventCapsuleHand)sender;

        CmdSetHandPosition(hand.Handedness, Positions.ToArray());
    }

    [Command]
    void CmdSetHandPosition(Chirality handedness, Vector3[] Positions)
    {
        //Debug.Log("CmdSetHandPosition");
```

```
        if(Positions == null)
        {
            Debug.Log("Positions was Null.");
            return;
        }
        if(handedness == Chirality.Left)
            for (int i = 0; i < Positions.Length; i++)
                syncedPositions_L[i] = Positions[i];
        else if(handedness == Chirality.Right)
            for (int i = 0; i < Positions.Length; i++)
                syncedPositions_R[i] = Positions[i];
    }

    public void InitHand()
    {
        if (_material != null)
        {
            jointMat = new Material(_material);
            jointMat.hideFlags = HideFlags.DontSaveInEditor;
        }

        if (_serializedTransforms != null)
        {
            for (int i = 0; i < _serializedTransforms.Count; i++)
            {
                var obj = _serializedTransforms[i];
                if (obj != null)
                {
                    DestroyImmediate(obj.gameObject);
                }
            }
            _serializedTransforms.Clear();
        }
        else
        {
            _serializedTransforms = new List<Transform>();
        }

        _jointSpheres_L = new Transform[4 * 5];
        _jointSpheres_R = new Transform[4 * 5];
        _armRenderers = new List<Renderer>();
        _cylinderTransforms = new List<Transform>();
        _sphereATransforms = new List<Transform>();
        _sphereBTransforms = new List<Transform>();
```

```
        createLeftSpheres();
        createRightSpheres();
        createLeftCylinders();
        createRightCylinders();

        //updateArmVisibility();

        _hasGeneratedMeshesL = false;
        _hasGeneratedMeshesR = false;
    }

    #region Model Creation for Remote Hands
    private void createLeftSpheres()
    {
        //Create spheres for finger joints
        //List<Finger> fingers = hand_.Fingers;
        for (int i = 0; i < 5; i++)
        {
            //Finger finger = fingers[i];
            for (int j = 0; j < 4; j++)
            {
                int key = getFingerJointIndex(i, j);
                _jointSpheres_L[key] = createSphere("Joint", SPHERE_RADIUS);
            }
        }

        mockThumbJointSphere_L = createSphere("MockJoint", SPHERE_RADIUS);
        palmPositionSphere_L = createSphere("PalmPosition", PALM_RADIUS);
        wristPositionSphere_L = createSphere("WristPosition", SPHERE_RADIUS);

        armFrontLeft_L = createSphere("ArmFrontLeft", SPHERE_RADIUS, true);
        armFrontRight_L = createSphere("ArmFrontRight", SPHERE_RADIUS, true);
        armBackLeft_L = createSphere("ArmBackLeft", SPHERE_RADIUS, true);
        armBackRight_L = createSphere("ArmBackRight", SPHERE_RADIUS, true);
    }

    private void createRightSpheres()
    {
        //Create spheres for finger joints
        //List<Finger> fingers = hand_.Fingers;
        for (int i = 0; i < 5; i++)
        {
            //Finger finger = fingers[i];
            for (int j = 0; j < 4; j++)
            {
```

```
            int key = getFingerJointIndex(i, j);
            _jointSpheres_R[key] = createSphere("Joint", SPHERE_RADIUS);
        }
    }

    mockThumbJointSphere_R = createSphere("MockJoint", SPHERE_RADIUS);
    palmPositionSphere_R = createSphere("PalmPosition", PALM_RADIUS);
    wristPositionSphere_R = createSphere("WristPosition", SPHERE_RADIUS);

    armFrontLeft_R = createSphere("ArmFrontLeft", SPHERE_RADIUS, true);
    armFrontRight_R = createSphere("ArmFrontRight", SPHERE_RADIUS, true);
    armBackLeft_R = createSphere("ArmBackLeft", SPHERE_RADIUS, true);
    armBackRight_R = createSphere("ArmBackRight", SPHERE_RADIUS, true);
}

private void createLeftCylinders()
{
    //Create cylinders between finger joints
    for (int i = 0; i < 5; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            int keyA = getFingerJointIndex(i, j);
            int keyB = getFingerJointIndex(i, j + 1);

            Transform sphereA = _jointSpheres_L[keyA];
            Transform sphereB = _jointSpheres_L[keyB];

            createCylinder("Finger_Joint", sphereA, sphereB);
        }
    }

    //Create cylinders between finger knuckles
    for (int i = 0; i < 4; i++)
    {
        int keyA = getFingerJointIndex(i, 0);
        int keyB = getFingerJointIndex(i + 1, 0);

        Transform sphereA = _jointSpheres_L[keyA];
        Transform sphereB = _jointSpheres_L[keyB];

        createCylinder("Hand_Joints", sphereA, sphereB);
    }

    //Create the rest of the hand
```

```
        Transform thumbBase = _jointSpheres_L[THUMB_BASE_INDEX];
        Transform pinkyBase = _jointSpheres_L[PINKY_BASE_INDEX];
        createCylinder("Hand Bottom", thumbBase, mockThumbJointSphere_L);
        createCylinder("Hand Side", pinkyBase, mockThumbJointSphere_L);

        createCylinder("ArmFront", armFrontLeft_L, armFrontRight_L, true);
        createCylinder("ArmBack", armBackLeft_L, armBackRight_L, true);
        createCylinder("ArmLeft", armFrontLeft_L, armBackLeft_L, true);
        createCylinder("ArmRight", armFrontRight_L, armBackRight_L, true);
    }

    private void createRightCylinders()
    {
        //Create cylinders between finger joints
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                int keyA = getFingerJointIndex(i, j);
                int keyB = getFingerJointIndex(i, j + 1);

                Transform sphereA = _jointSpheres_R[keyA];
                Transform sphereB = _jointSpheres_R[keyB];

                createCylinder("Finger Joint", sphereA, sphereB);
            }
        }

        //Create cylinders between finger knuckles
        for (int i = 0; i < 4; i++)
        {
            int keyA = getFingerJointIndex(i, 0);
            int keyB = getFingerJointIndex(i + 1, 0);

            Transform sphereA = _jointSpheres_R[keyA];
            Transform sphereB = _jointSpheres_R[keyB];

            createCylinder("Hand Joints", sphereA, sphereB);
        }

        //Create the rest of the hand
        Transform thumbBase = _jointSpheres_R[THUMB_BASE_INDEX];
        Transform pinkyBase = _jointSpheres_R[PINKY_BASE_INDEX];
        createCylinder("Hand Bottom", thumbBase, mockThumbJointSphere_R);
        createCylinder("Hand Side", pinkyBase, mockThumbJointSphere_R);
```

```
        createCylinder("ArmFront", armFrontLeft_R, armFrontRight_R, true);
        createCylinder("ArmBack", armBackLeft_R, armBackRight_R, true);
        createCylinder("ArmLeft", armFrontLeft_R, armBackLeft_R, true);
        createCylinder("ArmRight", armFrontRight_R, armBackRight_R, true);
    }

    private int getFingerJointIndex(int fingerIndex, int jointIndex)
    {
        return fingerIndex * 4 + jointIndex;
    }

    private Transform createSphere(string name, float radius, bool isPartOfArm = false)
    {
        GameObject sphere = new GameObject(name);
        _serializedTransforms.Add(sphere.transform);

        sphere.AddComponent<MeshFilter>().mesh = _sphereMesh;
        sphere.AddComponent<MeshRenderer>().sharedMaterial = jointMat;
        sphere.transform.parent = transform;
        sphere.transform.localScale = Vector3.one * radius * 2;

        sphere.hideFlags = HideFlags.DontSave | HideFlags.HideInHierarchy | HideFlags.
            ↪ HideInInspector;
        sphere.layer = gameObject.layer;

        if (isPartOfArm)
        {
            _armRenderers.Add(sphere.GetComponent<Renderer>());
        }

        return sphere.transform;
    }

    private void createCylinder(string name, Transform jointA, Transform jointB, bool isPartOfArm
        ↪ = false)
    {
        GameObject cylinder = new GameObject(name);
        _serializedTransforms.Add(cylinder.transform);

        cylinder.AddComponent<MeshFilter>();
        cylinder.AddComponent<MeshRenderer>().sharedMaterial = _material;
        cylinder.transform.parent = transform;

        _cylinderTransforms.Add(cylinder.transform);
```

```
        _sphereATransforms.Add(jointA);
        _sphereBTransforms.Add(jointB);

        cylinder.gameObject.layer = gameObject.layer;
        cylinder.hideFlags = HideFlags.DontSave | HideFlags.HideInHierarchy | HideFlags.
            ↪ HideInInspector;

        if (isPartOfArm)
        {
            _armRenderers.Add(cylinder.GetComponent<Renderer>());
        }
    }

    private Mesh generateCylinderMesh(float length)
    {
        Mesh mesh = new Mesh();
        mesh.name = "GeneratedCylinder";
        mesh.hideFlags = HideFlags.DontSave;

        List<Vector3> verts = new List<Vector3>();
        List<Color> colors = new List<Color>();
        List<int> tris = new List<int>();

        Vector3 p0 = Vector3.zero;
        Vector3 p1 = Vector3.forward * length;
        for (int i = 0; i < _cylinderResolution; i++)
        {
            float angle = (Mathf.PI * 2.0f * i) / _cylinderResolution;
            float dx = CYLINDER_RADIUS * Mathf.Cos(angle);
            float dy = CYLINDER_RADIUS * Mathf.Sin(angle);

            Vector3 spoke = new Vector3(dx, dy, 0);

            verts.Add((p0 + spoke) * transform.lossyScale.x);
            verts.Add((p1 + spoke) * transform.lossyScale.x);

            colors.Add(Color.white);
            colors.Add(Color.white);

            int triStart = verts.Count;
            int triCap = _cylinderResolution * 2;

            tris.Add((triStart + 0) % triCap);
            tris.Add((triStart + 2) % triCap);
            tris.Add((triStart + 1) % triCap);
```

```
            tris.Add((triStart + 2) % triCap);
            tris.Add((triStart + 3) % triCap);
            tris.Add((triStart + 1) % triCap);
        }

        mesh.SetVertices(verts);
        mesh.SetIndices(tris.ToArray(), MeshTopology.Triangles, 0);
        mesh.RecalculateBounds();
        mesh.RecalculateNormals();
        mesh.Optimize();
        mesh.UploadMeshData(true);

        return mesh;
    }
    #endregion

    #region Model Updating for Remote Hands

    public void UpdateHand()
    {
        //Update the spheres first
        updateLeftSpheres();
        updateRightSpheres();

        //Update Arm only if we need to
        if (_showArm)
        {
            //updateArm();
        }

        //The cylinder transforms are deterimined by the spheres they are connected to
        //Only update the cylinders if the meshes have been generated.
        //if(hasGenMeshesL && hasGenMeshesR)
            updateCylinders();
    }

    //Transform updating methods

    private void updateLeftSpheres()
    {
        if (syncedPositions_L.Count < 23) return;
        //Update all spheres
        for (int i = 0; i < 5; i++)
        {
```

```
            for (int j = 0; j < 4; j++)
            {
                int key = getFingerJointIndex(i, j);
                Transform sphere = _jointSpheres_L[key];
                sphere.position = syncedPositions_L[key];
            }
        }

        palmPositionSphere_L.position = syncedPositions_L[20];
        wristPositionSphere_L.position = syncedPositions_L[21];
        mockThumbJointSphere_L.position = syncedPositions_L[22];
    }

    private void updateRightSpheres()
    {
        if (syncedPositions_R.Count < 23) return;
        //Update all spheres
        for (int i = 0; i < 5; i++)
        {
            for (int j = 0; j < 4; j++)
            {
                int key = getFingerJointIndex(i, j);
                Transform sphere = _jointSpheres_R[key];
                sphere.position = syncedPositions_R[key];
            }
        }

        palmPositionSphere_R.position = syncedPositions_R[20];
        wristPositionSphere_R.position = syncedPositions_R[21];
        mockThumbJointSphere_R.position = syncedPositions_R[22];
    }

    private void updateArm()
    {
        Debug.Log("UpdateArm in Network Hand Controller was called , but not implemented");
        //var arm = hand_.Arm;
        //Vector3 right = arm.Basis.xBasis.ToVector3() * arm.Width * 0.7f * 0.5f;
        //Vector3 wrist = arm.WristPosition.ToVector3();
        //Vector3 elbow = arm.ElbowPosition.ToVector3();

        //float armLength = Vector3.Distance(wrist, elbow);
        //wrist -= arm.Direction.ToVector3() * armLength * 0.05f;

        //armFrontRight.position = wrist + right;
        //armFrontLeft.position = wrist - right;
```

```
        //armBackRight.position = elbow + right;
        //armBackLeft.position = elbow - right;
    }

    private void updateCylinders()
    {
        int start = hasGenMeshesL ? 0 : _cylinderTransforms.Count / 2;
        int end = hasGenMeshesR ? _cylinderTransforms.Count : _cylinderTransforms.Count / 2;
        for (int i = start; i < end; i++)
        {
            Transform cylinder = _cylinderTransforms[i];
            Transform sphereA = _sphereATransforms[i];
            Transform sphereB = _sphereBTransforms[i];

            Vector3 delta = sphereA.position - sphereB.position;

            if ((i<_cylinderTransforms.Count/2 && !_hasGeneratedMeshesL)||(i>=_cylinderTransforms
                ↪ .Count/2 && !_hasGeneratedMeshesR))
            {
                //Debug.Log("Update Cylinder:: Length: " + delta);
                MeshFilter filter = cylinder.GetComponent<MeshFilter>();
                filter.sharedMesh = generateCylinderMesh(delta.magnitude / transform.lossyScale.x
                    ↪ );
            }

            cylinder.position = sphereA.position;

            if (delta.sqrMagnitude <= Mathf.Epsilon)
            {
                //Two spheres are at the same location, no rotation will be found
                continue;
            }

            cylinder.LookAt(sphereB);
        }

        _hasGeneratedMeshesL = hasGenMeshesL;
        _hasGeneratedMeshesR = hasGenMeshesR;
    }

    #endregion
}
```

F.8 PinchDraw.cs

```
using UnityEngine;
using System.Collections.Generic;
using UnityEngine.Networking;
using System;

namespace Leap.Unity.DetectionExamples
{

    public class PinchDraw : NetworkBehaviour
    {

        [Tooltip("Each pinch detector can draw one line at a time.")]
        [SerializeField]
        private AbstractHoldDetector[] _pinchDetectors;

        [SerializeField]
        private Material _material;

        [SerializeField]
        private Color _drawColor = Color.white;

        [SerializeField]
        private float _smoothingDelay = 0.01f;

        [SerializeField]
        private float _drawRadius = 0.002f;

        [SerializeField]
        private int _drawResolution = 8;

        [SerializeField]
        private float _minSegmentLength = 0.005f;

        [SerializeField]
        private GameObject _LineParent;

        [SerializeField]
        private Material highlightMaterial;

        private int _layer;

        [SerializeField]
        private Tracker tracker;
```

```
private DrawState[] _drawStates;

public Color DrawColor
{
    get
    {
        return _drawColor;
    }
    set
    {
        _drawColor = value;
    }
}

public float DrawRadius
{
    get
    {
        return _drawRadius;
    }
    set
    {
        _drawRadius = value;
    }
}

void OnValidate()
{
    _drawRadius = Mathf.Max(0, _drawRadius);
    _drawResolution = Mathf.Clamp(_drawResolution, 3, 24);
    _minSegmentLength = Mathf.Max(0, _minSegmentLength);
}

void Awake()
{
    if (_pinchDetectors.Length == 0)
    {
        Debug.LogWarning("No pinch detectors were specified!  PinchDraw can not draw any
            ↪ lines without PinchDetectors.");
    }
}

void Start()
{
    _drawColor = GetComponent<NetworkHandController>().color;
```

```
        _layer = GetComponent<NetworkHandController>().PlayerNum + 7;
        foreach(var proxdetect in GetComponentsInChildren<ProximityDetector>())
        {
            proxdetect.UseLayersNotList = true;
            proxdetect.Layer = (1<<_layer);
        }
        _drawStates = new DrawState[_pinchDetectors.Length];
        _LineParent = GameObject.Find("Lines");
        for (int i = 0; i < _pinchDetectors.Length; i++)
        {
            _drawStates[i] = new DrawState(this, _LineParent, _layer);
        }
        //Only change the material color if we are the local player.
        if(GetComponentInParent<NetworkBehaviour>().isLocalPlayer)
            highlightMaterial.color = _drawColor;
    }

    void Update()
    {
        if (!isLocalPlayer) return;

        for (int i = 0; i < _pinchDetectors.Length; i++)
        {
            var detector = _pinchDetectors[i];
            var drawState = _drawStates[i];

            if (detector.DidStartHold)
            {
                string name = Guid.NewGuid().ToString();
                drawState.BeginNewLine(name);
                CmdBeginNewLine(i, name);
            }

            if (detector.DidRelease)
            {
                drawState.FinishLine();
                CmdFinishLine(i);
            }

            if (detector.IsHolding)
            {
                drawState.UpdateLine(detector.Position);
                CmdUpdateLine(i, detector.Position);
            }
        }
```

```
}

public void BreakConnection()
{
    //finish any lines that are currently being drawn.
    for (int i = 0; i < _drawStates.Length; i++)
        FinishLine(i);

    enabled = false;
}

public void MakeConnection()
{
    enabled = true;
}

[Command]
void CmdBeginNewLine(int drawState, string name)
{
    //Create a new Object that is tracked across the network to represent the line.
    tracker.LogStart(name,(Chirality)drawState);
    _drawStates[drawState].BeginNewLine(name);
    if (!enabled) return;
    RpcBeginNewLine(drawState, name);
}

void FinishLine(int drawState)
{
    tracker.LogEnd((Chirality)drawState);
    _drawStates[drawState].FinishLine();
    if (!enabled) return;
    RpcFinishLine(drawState);
}

[Command]
void CmdFinishLine(int drawState)
{
    FinishLine(drawState);
}

[Command]

void CmdUpdateLine(int drawState,Vector3 position)
{
    tracker.LogPoint(position, (Chirality)drawState);
```

```
        _drawStates[drawState].UpdateLine(position);
        if (!enabled) return;
        RpcUpdateLine(drawState, position);
    }


    [ClientRpc]
    void RpcBeginNewLine(int drawState, string name)
    {
        if (!isLocalPlayer)
            _drawStates[drawState].BeginNewLine(name);
    }

    [ClientRpc]
    void RpcFinishLine(int drawState)
    {
        if (!isLocalPlayer)
            _drawStates[drawState].FinishLine();
    }

    [ClientRpc]
    void RpcUpdateLine(int drawState, Vector3 position)
    {
        if (!isLocalPlayer)
            _drawStates[drawState].UpdateLine(position);
    }

    private class DrawState
    {
        private List<Vector3> _vertices = new List<Vector3>();
        private List<int> _tris = new List<int>();
        private List<Vector2> _uvs = new List<Vector2>();
        private List<Color> _colors = new List<Color>();

        private PinchDraw _parent;
        private GameObject _lineParent;

        private int _rings = 0;

        private Vector3 _prevRing0 = Vector3.zero;
        private Vector3 _prevRing1 = Vector3.zero;

        private Vector3 _prevNormal0 = Vector3.zero;

        private int _layer;
```

```
private GameObject _line;
private Mesh _mesh;
private SmoothedVector3 _smoothedPosition;

public DrawState(PinchDraw parent, GameObject lineParent, int layer)
{
    _parent = parent;
    _lineParent = lineParent;
    _layer = layer;

    _smoothedPosition = new SmoothedVector3();
    _smoothedPosition.delay = parent._smoothingDelay;
    _smoothedPosition.reset = true;
}

public GameObject BeginNewLine(string name)
{
    _rings = 0;
    _vertices.Clear();
    _tris.Clear();
    _uvs.Clear();
    _colors.Clear();

    _smoothedPosition.reset = true;

    _mesh = new Mesh();
    _mesh.name = "Line_Mesh";
    _mesh.MarkDynamic();

    _line = new GameObject(name);
    _line.transform.parent = _lineParent.transform;
    _line.transform.position = Vector3.zero;
    _line.transform.rotation = Quaternion.identity;
    _line.transform.localScale = Vector3.one;
    _line.AddComponent<MeshFilter>().mesh = _mesh;
    _line.AddComponent<MeshRenderer>().sharedMaterial = _parent._material;

    return _line;
}

public void UpdateLine(Vector3 position)
{
    _smoothedPosition.Update(position, Time.deltaTime);

    bool shouldAdd = false;
```

```
        shouldAdd |= _vertices.Count == 0;
        shouldAdd |= Vector3.Distance(_prevRing0, _smoothedPosition.value) >= _parent.
            ↪ _minSegmentLength;

        if (shouldAdd)
        {
            addRing(_smoothedPosition.value);
            updateMesh();
        }
    }

    public void FinishLine()
    {
        if (_line == null) return;
        _mesh.Optimize();
        _mesh.UploadMeshData(true);

        //Add a mesh Collider here so the proximity detector can work with it.
        _line.AddComponent<MeshCollider>().sharedMesh = _mesh;
        _line.layer = _layer;
        _line = null;
    }

    private void updateMesh()
    {
        _mesh.SetVertices(_vertices);
        _mesh.SetColors(_colors);
        _mesh.SetUVs(0, _uvs);
        _mesh.SetIndices(_tris.ToArray(), MeshTopology.Triangles, 0);
        _mesh.RecalculateBounds();
        _mesh.RecalculateNormals();
    }

    private void addRing(Vector3 ringPosition)
    {
        _rings++;

        if (_rings == 1)
        {
            addVertexRing();
            addVertexRing();
            addTriSegment();
        }
```

```
addVertexRing();
addTriSegment();

Vector3 ringNormal = Vector3.zero;
if (_rings == 2)
{
    Vector3 direction = ringPosition - _prevRing0;
    float angleToUp = Vector3.Angle(direction, Vector3.up);

    if (angleToUp < 10 || angleToUp > 170)
    {
        ringNormal = Vector3.Cross(direction, Vector3.right);
    }
    else
    {
        ringNormal = Vector3.Cross(direction, Vector3.up);
    }

    ringNormal = ringNormal.normalized;

    _prevNormal0 = ringNormal;
}
else if (_rings > 2)
{
    Vector3 prevPerp = Vector3.Cross(_prevRing0 - _prevRing1, _prevNormal0);
    ringNormal = Vector3.Cross(prevPerp, ringPosition - _prevRing0).normalized;
}

if (_rings == 2)
{
    updateRingVerts(0,
                    _prevRing0,
                    ringPosition - _prevRing1,
                    _prevNormal0,
                    0);
}

if (_rings >= 2)
{
    updateRingVerts(_vertices.Count - _parent._drawResolution,
                    ringPosition,
                    ringPosition - _prevRing0,
                    ringNormal,
                    0);
    updateRingVerts(_vertices.Count - _parent._drawResolution * 2,
```

```
                        ringPosition,
                        ringPosition - _prevRing0,
                        ringNormal,
                        1);
            updateRingVerts(_vertices.Count - _parent._drawResolution * 3,
                        _prevRing0,
                        ringPosition - _prevRing1,
                        _prevNormal0,
                        1);
        }

        _prevRing1 = _prevRing0;
        _prevRing0 = ringPosition;

        _prevNormal0 = ringNormal;
    }

    private void addVertexRing()
    {
        for (int i = 0; i < _parent._drawResolution; i++)
        {
            _vertices.Add(Vector3.zero);  //Dummy vertex, is updated later
            _uvs.Add(new Vector2(i / (_parent._drawResolution - 1.0f), 0));
            _colors.Add(_parent._drawColor);
        }
    }

    //Connects the most recently added vertex ring to the one before it
    private void addTriSegment()
    {
        for (int i = 0; i < _parent._drawResolution; i++)
        {
            int i0 = _vertices.Count - 1 - i;
            int i1 = _vertices.Count - 1 - ((i + 1) % _parent._drawResolution);

            _tris.Add(i0);
            _tris.Add(i1 - _parent._drawResolution);
            _tris.Add(i0 - _parent._drawResolution);

            _tris.Add(i0);
            _tris.Add(i1);
            _tris.Add(i1 - _parent._drawResolution);
        }
    }
```

```
            private void updateRingVerts(int offset, Vector3 ringPosition, Vector3 direction,
                ↪ Vector3 normal, float radiusScale)
            {
                direction = direction.normalized;
                normal = normal.normalized;

                for (int i = 0; i < _parent._drawResolution; i++)
                {
                    float angle = 360.0f * (i / (float)(_parent._drawResolution));
                    Quaternion rotator = Quaternion.AngleAxis(angle, direction);
                    Vector3 ringSpoke = rotator * normal * _parent._drawRadius * radiusScale;
                    _vertices[offset + i] = ringPosition + ringSpoke;
                }
            }
        }
    }
}
```

F.9 PointDetector.cs

```
using UnityEngine;
using System.Collections;
using Leap.Unity;

namespace Leap.Unity
{

    public class PointDetector : AbstractHoldDetector
    {

        protected override void ensureUpToDate()
        {
            //If we've already updated this frame no need to do it again.
            if (Time.frameCount == _lastUpdateFrame)
            {
                return;
            }

            //Set the last update frame to the current frame so we only update once per frame
            _lastUpdateFrame = Time.frameCount;


            _didChange = false;
```

```
        Hand hand = _handModel.GetLeapHand();

        if (hand == null || !_handModel.IsTracked)
        {
            changeState(false);
            return;
        }

        _distance = 0f;
        _rotation = hand.Basis.CalculateRotation();
        _position = (hand.Fingers[1].TipPosition).ToVector3();

        bool pointing = extendedFingerWatcher(hand, PointingState.NotExtended,
                            PointingState.Extended, PointingState.NotExtended, PointingState.
                                ↪ NotExtended, PointingState.NotExtended, 1, 1);

        if (IsActive && !pointing)
            changeState(false);
        if (!IsActive && pointing)
            changeState(true);

        if (IsActive)
        {
            _lastPosition = _position;
            _lastRotation = _rotation;
            _lastDistance = _distance;
            _lastDirection = _direction;
            _lastNormal = _normal;
        }
        if (ControlsTransform)
        {
            transform.position = _position;
            transform.rotation = _rotation;
        }
    }

    protected bool extendedFingerWatcher(Hand hand,
                                         PointingState Thumb,
                                         PointingState Index,
                                         PointingState Middle,
                                         PointingState Ring,
                                         PointingState Pinky,
                                         int MaximumExtendedCount,
                                         int MinimumExtendedCount)
    {
```

```
            bool fingerState = false;

            if (hand != null)
            {
                fingerState = matchFingerState(hand.Fingers[0], Thumb)
                    && matchFingerState(hand.Fingers[1], Index)
                    && matchFingerState(hand.Fingers[2], Middle)
                    && matchFingerState(hand.Fingers[3], Ring)
                    && matchFingerState(hand.Fingers[4], Pinky);

                int extendedCount = 0;
                for (int f = 0; f < 5; f++)
                {
                    if (hand.Fingers[f].IsExtended)
                    {
                        extendedCount++;
                    }
                }
                fingerState = fingerState &&
                    (extendedCount <= MaximumExtendedCount) &&
                    (extendedCount >= MinimumExtendedCount);
            }
            return fingerState;

        }

        private bool matchFingerState(Finger finger, PointingState requiredState)
        {
            return (requiredState == PointingState.Either) ||
                (requiredState == PointingState.Extended && finger.IsExtended) ||
                (requiredState == PointingState.NotExtended && !finger.IsExtended);
        }

    }

}
```

F.10 SpecialNetworkManager.cs

```
using UnityEngine;
using UnityEngine.Networking;

public class SpecialNetworkManager : NetworkManager {
```

```
    [SerializeField]
    private Canvas canvas;

    [SerializeField]
    private Camera mainCamera;

    // Use this for initialization
    public override void OnStartServer()
    {
        base.OnStartServer();

        if (canvas != null) canvas.gameObject.SetActive(true);
        if (mainCamera != null) mainCamera.gameObject.SetActive(true);

    }

    public override void OnStopServer()
    {
        base.OnStopServer();

        if (canvas != null) canvas.gameObject.SetActive(false);
        if (mainCamera != null) mainCamera.gameObject.SetActive(false);
    }
}
```

F.11 Tracker.cs

```
using Leap.Unity;
using System;
using System.IO;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.VR;

public class Tracker : NetworkBehaviour {

    [SerializeField]
    private bool log = false;

    private int numLogs = 1;
    private StreamWriter file;

    // Use this for initialization
```

```
void Start () {

}

public void SetLog(bool value)
{
    log = value;

    if(value)
    {
        try
        {
            file.WriteLine("Started Log# " + numLogs+ ".");
        }
        catch(Exception e)
        {
            Debug.LogException(e);
        }
    }
    else
    {

        try
        {
            file.WriteLine("Finished the Log# " + numLogs + ".");
        }
        catch(Exception e)
        {
            Debug.LogException(e);
        }

        numLogs++;
    }

}

public override void OnStartLocalPlayer()
{
    base.OnStartLocalPlayer();

    //Figure out what HMD this is and create a file name based on that
    string fileName = VRDevice.model + " Position.txt";
    CmdCreateFile(fileName);
}
```

```
[Command]
public void CmdCreateFile(string name)
{
    int playerNum = GameObject.FindGameObjectsWithTag("Player").Length;
    name = "Player " + playerNum + " Position Recordings.txt";
    Debug.Log("Trying to create the file: " + name);
    try
    {
        file = new StreamWriter("C:\\Position Data\\" + name, false);
        file.AutoFlush = true;
    }
    catch (Exception e)
    {
        Debug.LogError("Could not create the position log file.");
        Debug.LogException(e);
    }
}

public void OnDestroy()
{
    //Debug.Log("Destroying the tracker script.");
    if(file!=null)
        file.Close();

}

public void LogStart(string name, Chirality handedness)
{
    if (!log) return;

    try
    {
        file.WriteLine("Begin Line " + name + " from " + handedness.ToString("D") + " hand.")
            ↪ ;
    }
    catch (Exception)
    {
        Debug.Log("Error writing line start to file");
    }
}

public void LogPoint(Vector3 location, Chirality handedness)
{
    if (!log) return;
```

```
        try
        {
            //TODO: Fill in the time, and poition of the ball (and possibly which hand)
            file.WriteLine(Time.time.ToString("F3") + "\t" + handedness.ToString("D") + "\t" +
                ↪ location.ToString("F4"));
        }
        catch (Exception)
        {
            Debug.LogError("Error writing point to file");
        }
    }

    public void LogEnd(Chirality handedness)
    {
        if (!log) return;

        try
        {
            file.WriteLine("End Line from " + handedness.ToString("D") + " hand.");
        }
        catch (Exception)
        {
            Debug.Log("Error writing line end to file");
        }
    }

    public void LogDelete(string name)
    {
        if (!log) return;

        try
        {
            file.WriteLine("Line " + name + " was deleted");
        }
        catch (Exception)
        {
            Debug.Log("Error writing line delete to file");
        }
    }

}
```