University of Nevada, Reno

Virtual Reality Applications and Development

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Engineering

by

Christopher John Lewis

Dr. Frederick C. Harris Jr., Advisor

August, 2022

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

Christopher John Lewis

Entitled

Virtual Reality Applications and Development

be accepted in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Dr. Frederick C. Harris, Jr., Advisor

Dr. Sergiu M. Dascalu, Committee Member

Dr. Javad Sattarvand, Graduate School Representative

Markus Kemmelmeier, Ph.D., Dean, Graduate School

August, 2022

Abstract

Virtual Reality (VR) has existed for many years; however, it has only recently gained wide spread popularity and commercial use. This change comes from the innovations in head mounted displays (HMDs) and from the work of many software engineers making quality user experiences (UX). In this thesis, four areas are explored inside of VR. One area of research is within the use of VR for virtual environments and fire simulations. The second area of research is within the use of VR for eye tracking and medical simulations. The third area of research is within multiplayer development for more immersive collaborative simulations. Finally, the fourth area of research is within the development of typing in 3D for virtual reality. Extending from this final area of research, this thesis details an application that details more practical and granular details about developing for VR and using the real-time development platform, Unity.

Dedication

I dedicate this thesis to my father, John W.E. Lewis IV, "Papa Lew", for always being my greatest supporter and encouraging me to always push forward.

Acknowledgments

I would like to foremost thank my advisor and committee member Dr. Harris, for allowing me into his lab and letting me participate in research. I can't fully express what his guidance, trust, and knowledge for the last six years has done for me. I'd next like to thank the rest of my committee members, Dr. Dascalu and Dr. Sattarvand. Dr. Dascalu has been a part of my research and a steady hand of guidance for a number of years. I'm ever grateful for his passion that has partially led me into my current field of research. Dr. Sattarvand is new into my life, but his perspective in this thesis and his guidance in editing it is greatly appreciated. I hope to work with him more in the future. Next, I'd like to thank my Fiancée, Bianca, for her wit and the joy she brings to everyone around her, without her none of this would be possible. I'd also like to thank the authors of all of the many books and publications I've read over the years, for both scholastic and enjoyment purposes. There is nothing that sparks joy quite like a well written work. Lastly, I would like to thank all of the friends and coworkers I've worked with through the many years of research and schooling at the University of Nevada, Reno.

This material is based in part upon work supported by the National Science Foundation under grant number(s) 2019609 and IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As virtual reality (VR) continues to explode in popularity in corporate, education, entertainment, and research fields it is increasingly important to determine what VR can be used effectively for. It is well known that VR can increase a user's sense of immersion and presence in a virtual environment (VE). The benefits of VR come somewhat inherently from the medium itself, but also from a good user experience that finds it's roots in core human-computer interaction principles. This thesis explains portions of why VR is important, and what research has been done on it's capabilities.

VR's explosion of growth has left a distinct impression on households in the U.S. Reports [113] show that 23 percent of households have used or owned a VR headset. This figure is heavily dependent on generation, with Silent Gen, Boomers, and Gen-X having 4, 6, and 18 percent having used or owned VR headsets respectively. Gen-Y and Gen-Z have 38 and 45 percent used or owned VR headsets respectively. Monthly active users for VR in 2019 was, reportedly, at 43.1 million people while in 2020 this number shot up to 52.1 million people. It is estimated that by 2030, 23 million jobs will use AR and VR with healthcare, education, and blue-collar training being the most largely impacted.

The rest of this thesis is structured as follows: Chapter 2 provides background knowledge integral to understanding the material of this work including topics such as Virtual Reality, Software Engineering, and Human-Computer Interaction; Chapter 3 details a fire simulator application using Unity, REST data parsing, multiplayer, and

VR; Chapter 4 details another virtual reality application. This application utilizes 360° video, eye tracking, and VR to simulate a patient handoff; Chapter 5 details a VR multiplayer application that allows the users to simulate a medical procedure, called a patient handoff, using voice communication. The application also recorded the users interacting and their eye tracking information; Chapter 6 describes a novel approach to typing in VR using dictation and a "Drum-like Keyboard". This work also covers an associated user study on the typing methods; and Chapter 7 describes in detail certain processes for creating a functional game in VR, and the ambiance needed to make a semi-immersive virtual environment.

Chapter 2

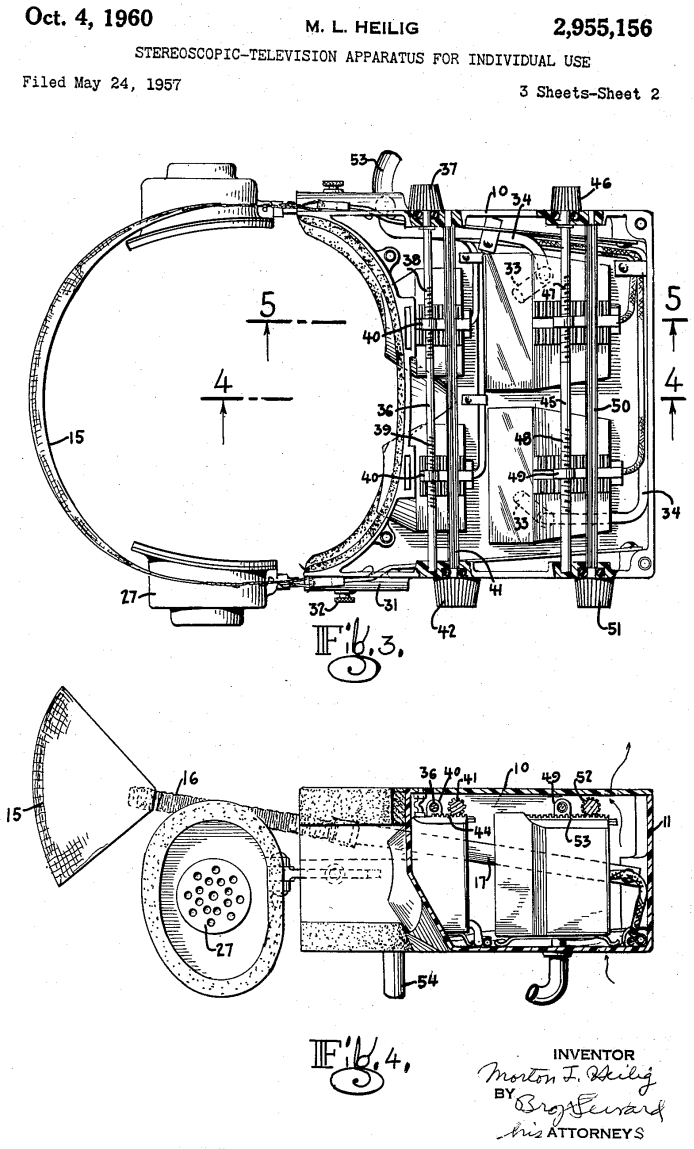
Background and Related Work

2.1 Virtual Reality

Virtual Reality (VR) as a topic is, perhaps, the most important aspect of this thesis to understand, as it is used and discussed in every chapter going forward. Thus, this section will be filled with the history of VR, modern research in VR, general development for VR, and some areas where VR still needs to improve.

2.1.1 History

While the current state of VR is dominated by commercially available head-mounted displays(HMDs) and various peripherals, an incredible amount of effort, time, resources, and research had to be invested first. VR hasn't always predominantly used HMD's, but it is where VR started. In 1960, a cinematographer named Morton Helig would receive a US patent for an invention that could show images, emit sounds, and emit air currents that could vary in velocity, temperature, and odor. This was the first known HMD and one of the diagrams submitted in the patent can be seen in Figure 2.1. This HMD was patented under the name of: "Stereoscopic-television apparatus for individual use" and is US patent number 2,955,156. Helig produced another notable advancement in VR called the Sensorama. The "Sensorama Simulator" was patented in 1962 and displayed 3D stereo video, stereo sound, aromas, wind, and had a seat that vibrated. The Sensorama's patent number is #3,050,870A and it's design can be seen in Figure 2.2. These inventions mark the emergence



Oct. 4, 1960 M. L. HEILIG 2,955,156

STEREOSCOPIC-TELEVISION APPARATUS FOR INDIVIDUAL USE

Filed May 24, 1957 3 Sheets-Sheet 2

Fig. 3.

Fig. 4.

INVENTOR
Morton L. Heilig
BY Brogdeen & Ward
his ATTORNEYS

Figure 2.1: The design of the first known HMD for VR, created by Morton Helig [37]

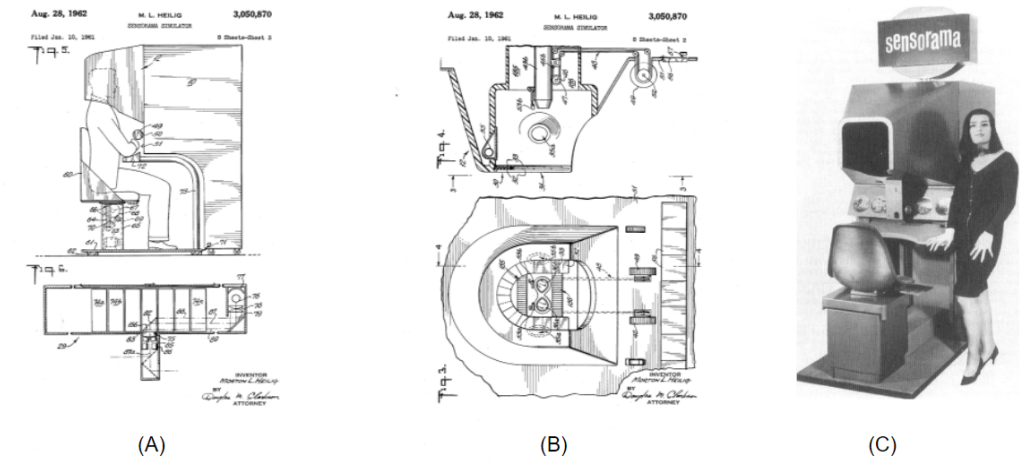


Figure 2.2: A & B: The patented design of the Sensorama simulator. C: The Sensorama after production [36]

of VR. Helig also wrote in his patent about the importance of this technology not only in a cinematographic sense, but also in: reducing hazardous situations/training for workers and the military; teaching devices to help education institutions; and multiplayer/social situations.

The next step in virtual reality quickly appeared after the Sensorama. This step came from Ivan Sutherland in the form of the "Sword of Damocles" seen in Figure 2.3. This invention was the first known HMD that could rotate the user's virtual field of view in tandem with how the user is physically moving their head. Dr. Sutherland's work and accomplishments are vast and could take quite a few pages of this thesis. Dr. Sutherland is credited as a pioneer of computer graphics. He received the Turing award for his PhD thesis, "Sketchpad", which was the first of it's kind to use a complete graphical user interface (GUI). It also influenced, if not created, modern graphical user interfaces (GUIs) and object oriented programming. Dr. Sutherland went on to create many other notable technologies and influencing many other notable students. Dr. Sutherland created "Sword of Damocles" in 1968 with a few of his students at Harvard University. The most notable students are: Bob Sproull, the former director of Oracle Labs and current adjunct professor at the University

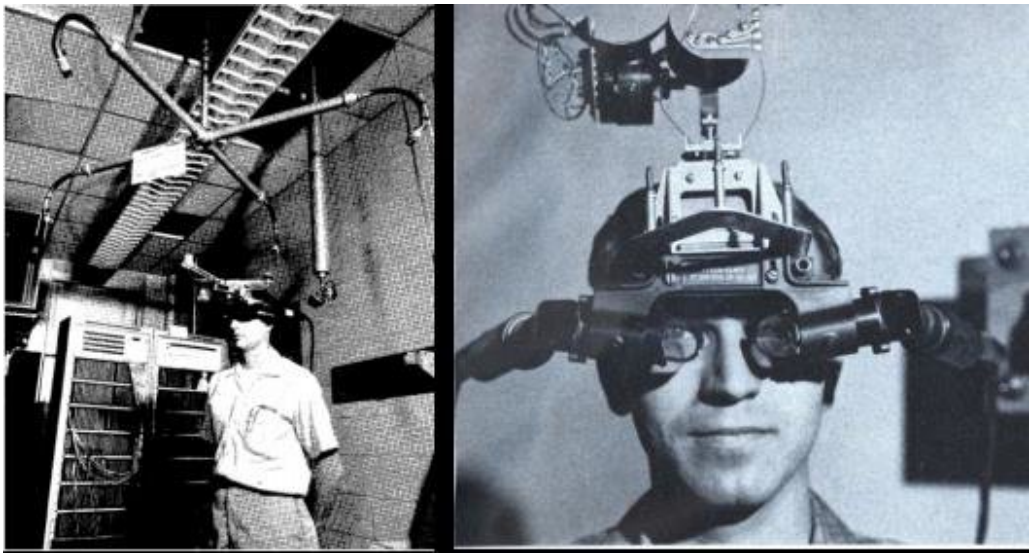


Figure 2.3: Ivan Sutherland's "Sword of Damocles" [97]

of Massachusetts Amherst; and Danny Cohen who adopted the terminology "endianness" for computing and has been inducted into the Internet Hall of Fame. The Sword of Damocles itself actually only refers to the mechanical tracking system and not the head-mounted display itself. The Sword of Damocles is also considered the first augmented reality system as the system was somewhat translucent.

From 1970 to 1990 most VR was developed for medical simulations, flight simulations, and military training purposes. A few notable inventions did occur during this time, however. In 1979, Eric Howlett created the Large Expanse, Extra Perspective (LEEP) optical system. LEEP had a wide field of view and was added to NASA's Ames Research Center in 1985. Next, Jaron Lanier founded VPL Research in 1985 where VR peripherals were being created. The most notable VR peripherals from VPL Research were the "DataGlove" and "DataSuit". The "DataGlove", as seen in Figure 2.4, was one of the first examples of a wired glove, which acts as an input device for human-computer interaction. These gloves generally mirror what the user is doing with their hands in virtual environments, though there has been some use for wired gloves to have a robot mimic what a human wearing the gloves is doing. This "DataGlove" was then licensed to companies to make entertainment

Figure 2.4: Someone using VPL Research's "DataGlove" [111]

related technology, most notably the “Power Glove”, which was used by Nintendo in their Nintendo Entertainment System. It didn’t sell well and users notoriously had a hard time with it’s controls and imprecision. The “DataSuit”, as seen in Figure 2.5 utilizes the “DataGloves” as well as a full body suit that is filled with sensors that can measure the movement of arms, legs, and the torso.

Figure 2.5: VPL Research’s DataSuit displayed at the Nissho Iwai showroom in Tokyo, Japan [112]

The 1990’s saw some of the biggest changes to VR since it’s inception and the seminal systems by Dr. Sutherland. One of the largest issues with VR before this point was it’s cost. None of these headsets were available for commercial use and they were all generally geared towards large institutions like military, medicine, or academia. In 1991 Sega announced Sega VR, which never made it to release. That same year,

Virtuality launched the first mass-produced multiplayer VR systems. These systems were created for the use in VR arcades and had a cost of $73,000 per system. While not quite commercially available to the end user, this shows a distinct increase in the use of VR for entertainment and non-industry use. Again in 1991, the next large step in VR was taking place in academia.

The Cave Automatic Virtual Environment (CAVE) was a PhD thesis created by Carolina Cruz-Neira [19]. Daniel J. Sandin, a professor emeritus at the University of Illinois at Chicago, and Thomas A. DeFanti, a professor at the University of Illinois at Chicago, are also credited with creating the CAVE. The CAVE can come in quite a few forms, but the most complete CAVE is a six-sided fabric lined room using one projector and one mirror to light each side of the room, from the outside, with features from a simulated virtual environment. One example of a four-sided CAVE can be found in Figure 2.6. While this figure doesn't depict a complete six-sided CAVE, the same principles apply for any number of sides on a CAVE. This figure also shows the tracking cameras which allow the user inside of the CAVE to move around and interface with the virtual environment in various ways. Originally this

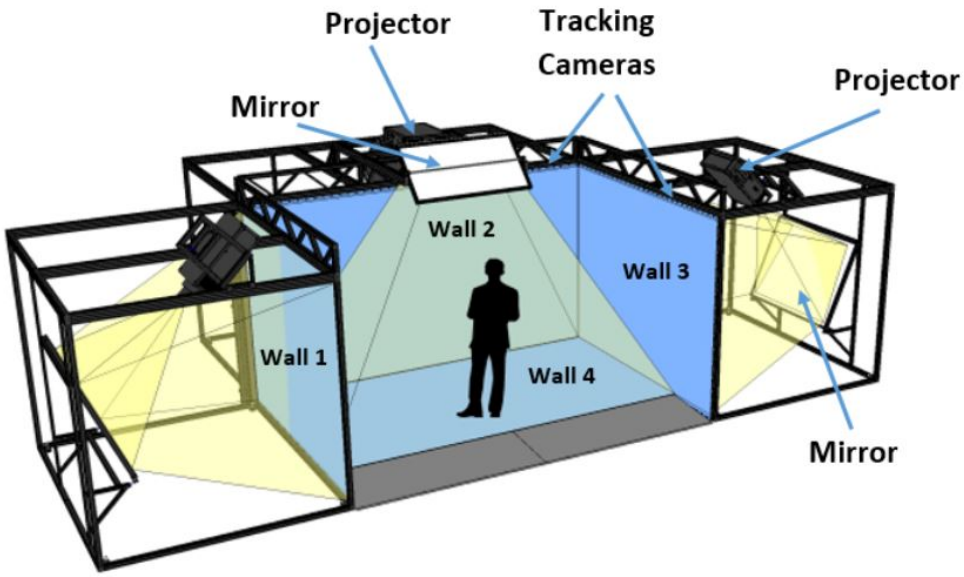


Figure 2.6: Annotated diagram of a four-sided CAVE system with mirrors from NASA's GRUVE Lab [72]

technology used electromagnetic sensors to track movements, but has since moved to infrared cameras to eliminate interference common to electromagnetic sensors. This technology has been widely adopted and you can find CAVE systems, despite their high price and long setup times, at many universities and research facilities. This technology has also evolved, as of 2012, to produce the CAVE2 [12], which is based on LCD panels rather than full projection as seen in Figure 2.7. The CAVE2 was produced by the Electronic Visualization Laboratory (EVL) at the University of Illinois, Chicago. The two most significant aspects of the CAVE2 system are that it's cost is considerably lower than the CAVE system, and it allows for a more spherical shape to the environment, which allows for much greater realism. The CAVE2 also boasts ten times the 3D resolution of the original CAVE.

Figure 2.7: Depiction of the CAVE2 from EVL at UIC [12]

Many other inventions happened after the CAVE to produce 3D graphics, but no real progress in VR devices happened until 2010, aside from large scale VR hardware in the use of theaters and amusement rides. In 2010, the prototype for the Oculus Rift was released. This headset boasted a 90-degree field of view, which wasn't previously seen before in HMD based VR. The Oculus Rift would then be shown off at E3 and Facebook, now Meta, ended up buying it for three billion USD. Meta and Oculus later

got sued by Zenimax, over the Oculus on grounds of dissemination of company secrets, who won after Meta settled out of court. The next important step in HMD innovation was done by Valve who discovered, and freely shared, low-persistence displays which make smear-free HMDs for VR possible. This technology would then be adopted by all HMD manufacturers going forward. In 2014 Sony announces Playstation VR (code name Project Morpheus). In 2015 the HTC Vive would be announced and it would use tracking technology which utilized "Lighthouses" or "base stations" that use infrared light for position tracking of the VR headset and controllers. In 2015, Google announced Google Cardboard which would bring VR to a brand new audience by utilizing smartphones for VR. Going forward, almost every large tech company either had a VR HMD released or a VR/AR group at the company.

In the current era, each of these HMDs are starting to carve their own niches in the VR space. Despite these niches, most HMDs can be categorized based off of their tracking method and connection type. Tracking methods are either outside-in or inside-out. Outside-in tracking is where the HMD, and other peripherals, are being tracked by outside sensors. Some HMDs that offer this tracking are the Oculus Rift, Valve Index, HTC Vive Pro, HTC Vive Pro Eye, HTC Vive Pro 2, and the PS VR system. Inside-out tracking is where the HMD is tracked via integrated sensors that detect changes in the position of objects in the environment. This type of tracking can be done either with or without markers placed in the room. Some HMDs that offer this tracking are the HTC Vive Cosmos, Microsoft HoloLens, and Meta Quest 2. The HTC Vive Cosmos Elite is a particularly interesting VR headset due to the fact that it allows for both inside-out and outside-in tracking due it's replaceable face plates. Connection types for HMDs mean that the HMD is either connected to the PC to be able to work, or it has a standalone system that allows the headset to work without a PC attached. Most headsets are not standalone, some that are standalone are the: Meta Quest, Pico Neo 3 Link, HTC Vive Focus 3, and HTC Vive Flow. Currently, the product line with the most flexibility and diversity is done by HTC Vive, especially now that Oculus/Meta no longer produce any HMD besides the Meta

Quest 2.

2.1.2 Current Research

Most current research is in the software and tools for VR since creating unique and usable headsets is expensive and is already being done by large manufacturers. There is quite a bit of research dedicated to reducing or better understanding virtual reality sickness (VR sickness). There are also quite sizable projects designing applications to train or educate individuals/groups.

2.1.2.1 VR Sickness and Health Risks

VR sickness, which is also called cybersickness, closely resembles motion sickness in terms of symptoms. This sickness comes with a few known symptoms: Nausea, balance disorder, and vomiting. VR sickness is generally understood to be caused by a disconnect between what our senses are telling us and what is actually happening. This is called sensory conflict theory and it has been used to understand motion sickness for many years, though there are many theories relating to this topic. An example of what sensory conflict theory is presenting: If a person is in a car and looks out a window, they may get motion sickness due to the fact that their eyes see a fast moving landscape, but they personally are not moving. This would explain why some people's motion sickness gets better when they look at objects in the environment that are further away, and are thus not moving as fast due to parallax. There is also a notable conflict in the literature between deciding if gender has a role to play in VR sickness with more recent research dictating that there is no significant difference [88, 114], while later research explains that women are more susceptible to VR sickness [52, 79]. It is not clear why this conflict in the literature exists, but due to the almost ten years of time difference between the studies it could be quite a number of things.

Though not currently being researched very heavily, it has been shown that VR can be problematic for a user's sense of presence and can even induce dissociation [1]. This work shows the symptoms of depersonalization and derealization had a signif-

icant increase (4.9% - 14.5%) in their thirty participants when exposed to a virtual environment in VR. These symptoms exist to some extent in every individual, but in both the cases of participants with high and low amounts of these symptoms/feelings already, there was a significant increase in these symptoms/feelings. Due to these findings: the more realistic a virtual environment, the more careful developers need to be in showing a user unsettling and graphic things as they could severely impact their users world view and sense of self outside of the real, objective, reality. Quite a bit more research should be put into this topic, in particular, if the dissociation issue should be listed as a part of VR sickness, meaning that they effect the same people in the same way. It would also be interesting to see if a number of factors change the amount of disassociation in the participants: the amount of realism, resolution of the environment, interactions available, multiplayer, ambient sound or background music, HMD differences, and/or locomotion techniques. It also isn't clear how long these symptoms last.

Lastly, there is another important health risk that users and developers of VR must be aware of. HMDs can be quite heavy at around 600g or 1.3lbs, so using them for extended duration and at a fast movement rate can be slightly dangerous. If the user's back, neck, or spine are sore do not continue to use the HMD. This disclaimer is brought about due to a very recent report of a German VR gamer breaking their C7 neck vertebra while playing VR [8]. This is the first ever case of a VR induced stress fracture, despite the sixty years of HMDs before this point. More cases are likely to pop up due to VR's growing proliferation, however.

2.1.2.2 Locomotion

Due to the fact that VR is a fairly new technology, a lot of work is being put into making it more usable and user friendly. One issue for VR is in typing, which will be covered extensively in Chapter 6. Another issue is in locomotion. Physical locomotion, which is the process of physically moving and having the VR system track and display this, by default this isn't really that practical. Outside-in tracking causes this

type of locomotion to be very limited due to the need to stay in range of a sensor. Inside-out tracking makes this locomotion a bit more plausible, but still relatively useless by itself as you're still tethered to a PC. The major downside to this locomotion technique not being usable is in the fact that physical bipedal walking comes the most naturally to humans and thus it makes VR sickness less likely. This is why redirected walking [83] was developed in 2001. Redirected walking rotates the environment around the user slowly and, almost, imperceptibly. The principle relies on the fact that humans will naturally auto-correct their movement in order to navigate through an environment. As they naturally auto-correct their physical rotation in order to direct themselves to an objective in their virtual environment, they create a curved physical path, thus increasing their available play space without even realizing it, as they only think they've walked in a straight path in the virtual environment. Many other publications have claimed they've improved on this concept using specific algorithms and machine learning [7, 55, 100], but the core concept remains the same.

Another physical locomotion technique is walking in place. This technique allows walking by having the user bring their legs up in a walking-like motion, but not actually move to anywhere physically. This technique is seldom used in favor of some of the other techniques listed below. There is an iteration of this technique that is used regularly, which is held-in-place walking or gait-negation. This technique utilizes a third party peripheral to hold the user in place as they walk or run. Two of these peripherals are the "Virtuix Omni-directional VR Treadmill" and the "Virtuix Omni One VR Treadmill". Both utilize a very slippery surface, to reduce friction of feet moving, and trackers attached to the user's shoes to detect movement. The way these two treadmills differ is in their holding mechanisms. The Omni-directional VR treadmill, as seen in Figure 2.8, uses a ring around the user that the user can lean against to move towards a virtual location. The ring itself is set to a specific height for each user before they get into the VR environment. As the user leans they slip and can walk or run forward towards that virtual location. The "Omni One" [106], as seen in Figure 2.9, holds the user by strapping them into a full vest that is suspended at

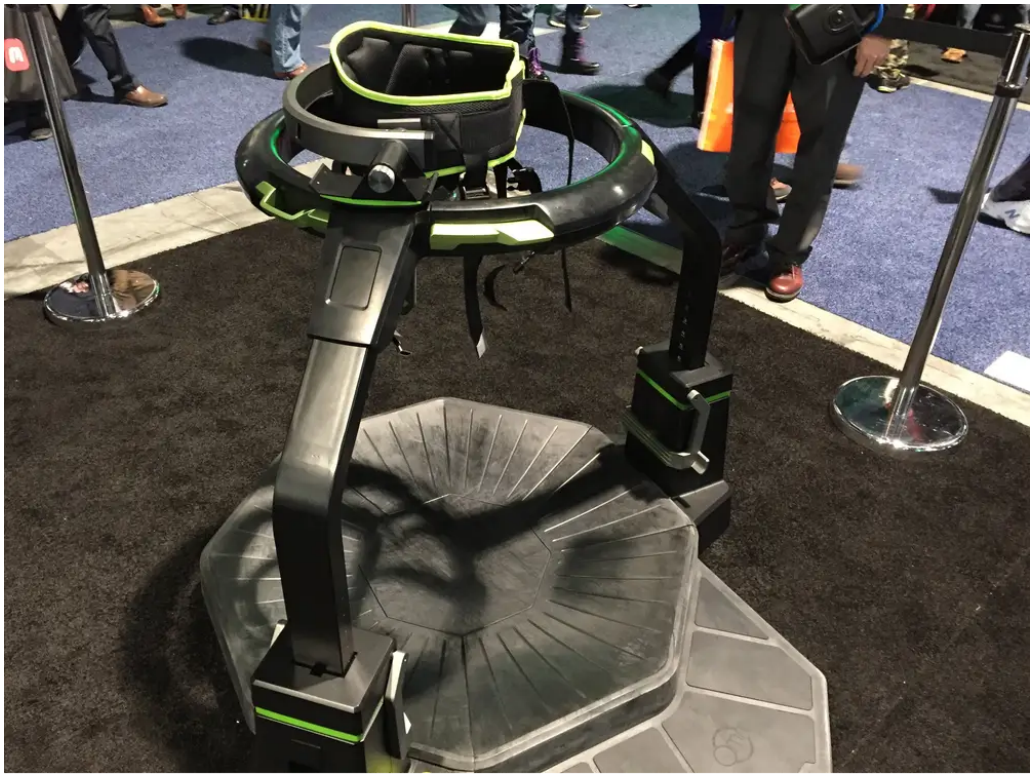


Figure 2.8: The Virtuix Omni-directional VR treadmill [103]

a given height. This particular model reportedly allows the user to jump and crouch as well, something that the previous model did not. This treadmill is not currently out on the market yet, but they do have demos that make this treadmill seem very interesting for research going forward.

There are quite a few other locomotion techniques. Joystick walking is where the user utilizes a joystick or a trackpad of some sort to move their avatar in a given direction. This approach can have six degrees of freedom due to the nature of the controllers. This approach also has a significant risk of VR sickness due to the fact that the user's perspective is moving, but the user themselves are not physically moving. This technique doesn't require the use of trackpads or joysticks, but some continuous actuator is required as well as some sort of vector. For example, the trigger on a controller could be the actuator and the rotation of the controller could give the angle of the vector, while the magnitude is fixed programmatically.

Teleportation based movement can come in quite a few forms. The three most

Figure 2.9: The new Virtuix Omni One VR treadmill [106]

prevalent forms are what we'll call: direct teleportation, preset teleportation, and avatar movement-based teleportation. Direct teleportation, in this case, refers to teleporting directly to the location that the user's cursor is. The cursor will generally be shot out from the tip of the user's controller and fall rapidly until it collides with the floor. This then creates a target on the ground that the user can then teleport to by pressing a button. This is by far the most common teleportation method and it is the default inside of the "SteamVR Home" [43] application (the default VR launcher on a PC). "SteamVR Home" also uses preset teleportation. This teleportation allows the user to teleport only to specific location in the environment. These locations are either single points that allow the cursor to snap directly to them, or they are larger areas that utilize a small amount of direct teleportation. The combination of direct and preset teleportation results in areas that the user can teleport inside of, but are still defined by the developers. "SteamVR Home" utilizes all of these teleportation methods by having a preset area that the user can teleport inside of and specific points in that area that the user can teleport to in order to select and change specific aspects of the virtual environment. The last teleportation method, avatar movement-based teleportation, is quite interesting. This teleportation method is the least common, but it is a very novel approach. This approach aims to solve a social VR problem, where teleportation generally feels very off putting to the people around the user teleporting. This method animates the user's avatar and makes it walk to the location the user's cursor is focused, while keeping the user's camera fixed in their current position. When the avatar makes it to their desired location, the user can then teleport their camera to that location. This means that the user temporarily sees in third person and can watch their avatar walk to a location. This locomotion technique can be seen in the application, "VR Chat" [49], which is a social VR platform. It is important to note that none of the teleportation methods can move with the six degrees of freedom that the joystick allows, but they also have significantly less risk of VR sickness associated with them.

2.1.2.3 Education and Training

Education in VR has always been a key field where VR shines. In recent years, training in diversity, equity, and inclusion (DEI) have been large topics of discussion. Virtual reality has been used to further this type of training to make users more engaged and present rather than simply for compliance. Technology in this specific area has been focusing on social VR, 360° video, and speech recognition, to name a few. One DEI VR application [86] attempted this type of training and met with resounding success from participants. The users in this study responded to the training's questionnaire and the researches released the following data points: 90.8% felt moderately to completely engaged; 60.5% reported feeling somewhat present of very present during the VR experience; 94.7% of respondents agreed or strongly agreed that VR was an effective tool for enhancing empathy; 85.5% agreed that the VR experience enhanced their own empathy toward racial minorities; 18.4% reported discomfort in VR; and 67.1% stated that they believed the VR experience would change their approach to communication. These survey results are overwhelmingly positive, and they validate what corporations have started to do already, which is to train their employees in DEI using VR.

Training in VR not only has the benefit of generally being more immersive and causing trainees to feel more present in their training, it also has the benefit of allowing accurate and realistic simulations and training for dangerous situations virtually. From fire simulations based in a user's home town to training for mining and construction related difficulties, virtual reality training can help save many lives and prepare individuals more accurately than many other techniques. This is also true for training in VR for medicine. One paper, depicting the results of a randomized, double-blinded study on VR training for the operating room in gallbladder surgery [91], found that VR training was 29% faster than a traditional approach and that non-VR-trained residents were nine times more likely to transiently fail to make progress. Non-VR-trained residents were also five times more likely to injure the gallbladder or burn non-target tissue. Mean errors were also six times less likely to occur for the VR-

trained group.

Indeed, the effectiveness of VR training and education can be very worthwhile for quite a few fields. However, this doesn't mean that VR is correct for every situation and field. To give more understanding as to when to use VR for training, a paper titled, "Reasons to Use Virtual Reality in Education and Training Courses and a Model to Determine When to Use Virtual Reality" [78] shows insight into this problem. This paper advocates for the use or consideration of VR when: Simulations could be used; teaching or training using the real thing is dangerous, impossible, inconvenient, or difficult; a model of an environment will teach or train as well as the real thing; interacting with a model is at least as motivating as with the real thing; Travel, cost, and/or logistics of gathering the class are too high compared to using VR; shared experiences and environments are important; the creation of the environment or model is part of the learning objective; information visualization is needed; a situation needs to be made to feel real; improving perception on specific objects; developing participatory environments and activities that can't exist in the real world; teaching tasks that involve dexterity or movement; a want to make learning more interesting and fun; accessibility for disabled people; or where mistakes in the real world would be devastating or demoralizing. This paper also describes the reasons not to use virtual reality for training/education: no substitution is possible; interactions with real objects is necessary; using a VE could be physically or emotionally damaging; using a VE can result in the confusion between reality and the VE; or VR is too expensive given the learning outcome.

2.1.3 Areas for Improvement

VR has a number of areas that still need research. One issue that will be a problem for future VR applications is user verification. Currently, most verification is done via generic username and password in a PC environment. This is secure up until the user puts on the HMD. Once the user puts on the HMD, it could be anybody underneath the headset. This has caused a number of research papers to come out

on how to verify users in VR via movement and how to track and visualize movement in VR [54, 66, 76]. There has also been interesting work on signature validation using data gloves [89]. Despite these verification techniques, more research needs to be put into the continuous verification of users due to the fact that users can take their headset and swap with another user at any point in an application. This will be particularly relevant and important when VR is more widely adapted in training, education, and the workplace.

Data gloves are integral for the future of VR due to their versatility and natural movement. The hardware itself does need a bit of work, in most cases on it's tracking capabilities. This is particularly true for the actuation and range of movement of the hand and fingers. Largely, the hardware is usable. The main issue with data gloves is on the software side of things. Gesture recognition is possible with these gloves [17], but out of the box they recognize few, if any, gestures. Developers who use these gloves also, generally, have to develop for each glove manufacturer's SDK. More research and development must be put into generalizing these SDKs so that developers can program for data gloves as a whole, rather than individual types of gloves.

Implementing these abstraction layers between VR technology and VR applications is incredibly important, not just for data gloves, but also HMDs. The industry is moving towards this abstraction layer called, "OpenXR" which is an application interface that most HMDs use. Oculus is currently the only popular HMD that isn't fully integrated with OpenXR yet, but the company has decided that after August 31st, 2022, all developers will need to switch to developing with OpenXR and old SDKs will be unsupported. Both Unity and Unreal Engine 5 have support for OpenXR and each have a VR template supporting the features in OpenXR.

VR templates are a good step forward in VR development; however, most have very little features actually associated with them. This needs to change in order for VR to be easier to develop for. These templates do not have support for many interactions and peripherals associated with VR development, and can often not

have many locomotion features, selection features, 3D models specific to VR, and multiplayer. These are all features that need to be specifically tailored for VR and would help developers immensely in creating applications quickly.

Typing in VR needs more research and development. Typing has always been a much slower process in VR than in traditional interfaces. This work aims to help this issue, but it is not a complete solution by far. Natural typing is immensely important due to the fact that most applications require it. It's part of a large population's day-to-day activities and VR should be able to accommodate something like it easily in order for it to grow into a useful day to day platform in work, education, training, and entertainment.

As with any 3D environment, VR needs to have better and more realistic graphics. Photogrammetry can go a long way, as described in Chapter 5, but it is a long way away from perfect. Even in a 3D scanned and photogrammetry created room, shadows and textures can create an unnatural and uncanny valley type feeling. The world around the user can look almost real, but there are slight differences that clearly make it look like good CGI and not an actual environment. Most people could not put it into words as to why the scene looks off, but they do notice that it isn't correct.

Overall, VR has quite a few issues to iron out before it can widely be used, but the potential of it is truly remarkable. There may be a world in the future where education is done via group experiences in VR to participate in science experiments and minutely control the intricacies of the simulation and playback speeds. Where training is done in VR to prepare workers for any situation. Where desk jobs and programming are done in your own home, yet you're virtually at the office and can still interact socially with your coworkers. Where joggers are virtually running with their friends all over the world, yet jogging physically alone in their own neighborhood. Truly, VR/AR/MR/XR have an incredible amount of potential in connecting us socially, educating the masses in new, unique, and effective ways, entertaining in a unique and interesting medium, and creating safe and efficient work places.

2.2 Human-Computer Interaction

Human-Computer Interaction (HCI) is a very broad subject with overlaps in user experience design, user interface design, human factors engineering, computer science, cognitive science, and many other fields.

2.2.1 Context-Aware Computing

Context-Aware Computing utilizes situational and environmental information to predict and anticipate needs and tasks. This type of computing is very popular in Internet of Things and Internet of Everything type systems. In the seminal paper for Context-Aware Computing [90], this type of computing application is categorized into four different types: proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions. Proximate selection is the technique where context-aware computers can select objects based off of their proximity to the computer. This type of context-awareness is seen most commonly with network-enabled printers and with location-based searches. Most commonly used is, currently, location-based searches where users identify an object that is "near me". For example, users may search "hikes near me", "restaurants near me", or "apartments near me". Printers are location-aware due to the fact that the OS can automatically select the printer in the same IP range as the device that requires a printer. Automatic contextual reconfiguration describes the process of automatically adding or removing components based off of certain contexts. The paper references adapting to the location a person is in by adding or removing components like a whiteboard that's in use in a room. This type of context-aware computing is very underused in the current year. In virtual environments, this is done regularly with virtual objects though not in a context-aware way. Where if you virtually move into one location, you can see different objects and how the other users were interacting with them. Contextual information and commands describes a system that can change based off of user actions changing in different environments. This means that

as a person might move from room to room, they might receive different information or give certain commands. The information and the commands might even stay the same, but they might do different things. For example, in a house full of unused displays, a user might want to press a button and have their workstation move to the closest display. The button stays the same, but the action itself changes based off of the proximity. If a user moves from one room to another, network drives may differ with specific notes in each. For example, if a user moves into their office they might see a network drive filled with work notes, their kitchen might have information on how to operate a coffee maker, etc. Context-triggered actions are rules to tell the context-aware system how to adapt. These context-triggers allow the system to do a multitude of different functions and changes. These are essentially event-based systems that can call other systems or trigger a cascade of other functions. These types of context-aware computing are not the full extent of it's capabilities anymore, but these categorizations do encompass a large section of them.

There are many examples of Context-Aware Computing in the modern day. One example is in Conversational Agents (CAs), software that can identify speech and respond to it in certain ways. CAs are embedded into many of the common technology people use, from Microsoft's Cortana which is integrated into the Windows operating system, to Apple's Siri and Samsung's Bixby. Most large brands have answering machines that can utilize natural language processing to figure out needs of their users and chatbots are extremely common for online help. All of these are CAs that are forms of Context-Aware Computing. CAs, particularly in smartphones, allow users to quickly search for information, open specific applications, and run specific useful functions. For examples on all of these features: users can look up the weather in their area for the next week, they can open up the camera app without having to navigate to it, and they can turn on or off "smart appliances" just with their voice and a conversational agent. Despite the usefulness and availability of these CAs, many users don't like using them and can often be frustrated with the results the CAs bring forward. This is further illustrated by a work titled, ""Like Having a Really

Bad PA": The Gulf between User Expectation and Experience of Conversational Agents" [61]. This paper illustrates Donald (Don) Norman's, "Gulf of Evaluation and Gulf of Execution", which was popularized in the original, "The Psychology of Everyday Things" [74], which was revised as, "The Design of Everyday Things" [73]. The gulf of execution is in the difference between what a user intends to do with a system compared to what the system allows them to do and to what degree of execution does the system do those things. The gulf of evaluation is in how easily a user can determine, or evaluate, the current state of the system. It is postulated by Norman that lowering these gulfs in any system will increase user experience. CAs were found to have very large gulfs in both execution and evaluation. CAs often didn't execute in the way users expected and users had to actively adjust not only their expectations for the system, but they actively had to learn what the system was capable of through repeated interaction. Not only is there practically no way to evaluate CAs, but they generally don't have much of a user facing UI in general. The most amount of evaluation the system gives is that some CAs display when/if the user is talking and verbally tell the user that it doesn't understand what the user is asking for. Furthermore, each CA has a different level of capabilities, so it is never initially clear, from the user's perspective, how the user needs to communicate to the CA to get what they want. It also isn't clear if the system even has the capability to answer the user's question at all, let alone to a significant level. The paper also mentioned the conversational differences between what the CA is able to understand and what the CA replicates. CAs can often use humorous language or communicate in a somewhat conversational way, while the language the CA requires the user to communicate in is not conversational and often boiled down to simple language and concepts.

2.2.2 Perceptual Interfaces

Perceptual Interfaces allow interaction with a computer through the use of senses, motion, or speech. This type of interface includes gesture recognition, verbal commands,

motion-based interaction, and many other new and exciting types of interaction. Exciting work has been done in brain-machine interfaces (BMIs), which is it's own type of interface as well as a perceptual interface. Researchers have made interfaces that allows a mouse to move a mouse cursor [15] and, among other exciting research, allowed a tetraplegic [9] patient control an exoskeleton with a somewhat large amount of success. Among other BMIs, electroencephalogram (EEG) headsets have shown promise in examining the electrical signal output by a human brain for stress [82], focus [58], and even brain disorders, like epilepsy [120].

Gesture recognition is the process of recognizing meaningful movements. Gesture recognition has been key in much research, from sign language to medical rehabilitation and into virtual reality. While gesture recognition largely relates to hand gestures, there are also body [95] and even vision-based gesture techniques [116]. A relevant paper that involves hands and feet gesture recognition, as well as verbal commands, is focused on navigating images in an operating room (OR) [84]. Navigating images is important for surgeons in the OR. Currently, many ORs have the doctors giving verbal cues to non-sterilized staff to navigate images for them, due to the fact that the surgeons themselves have sterilized hands and can't operate technology while operating on a patient. The authors created touch-free interactions using the HTC Vive and Leap Motion Controllers. This allowed the surgeons to use foot and hand gesture controls as well as verbal commands to navigate through images. In a user study showcasing the effectiveness of these approaches, the authors found that verbal commands completed tasks quicker, were rated most favorably by the participants, and had higher overall usability based on the system usability score (SUS). While the absolute numbers reflect this, the authors found no significant difference across the techniques. Surgeons also generally commented positively on the VR system and interaction techniques.

2.2.3 Social Computing

Social Computing is the interactive and collaborative behavior between people using computers. Much public discourse has been raised over social computing in recent years. From social media, where after the recent acquisition of Twitter by Elon Musk, many individuals are questioning what roles social media play in our society. Do social media platforms have a requirement to our society to protect individuals from dangerous topics? Do platforms have a requirement to our society to block or identify misinformation? Do social media platforms have a requirement to our society to reach out and develop algorithms to reach out to those displaying suicidal thoughts? Many questions like these are being researched. One research paper [14] found a statistical link between individuals who engage in mental health discourse on social media who then transition to suicidal ideation in the future. The paper also found specific markers could characterize these shifts and could identify users who were likely to undergo these transitions using those markers. The authors found that they could also distinguish between users likely to transition and users who wouldn't.

Social computing can also bring users together. Teleconference software can bring people together from many miles away for work, school, and relationships. It is easier than ever to interact with those you care about, as well as develop meaningful relationships online.

One research paper [69] found that social VR, another form of social computing, can help foster and continue communication to both brand new relationships and older relationships that were originally in real life. This paper found some overall themes that are really indicative of where social VR is in terms of research and usability: Users tend to mimic real-world behavior, this is one of the truly great things about social VR as a whole. Due to the voice communication, physical models, and associated names in VR, users of VR can often be authentic, even without a large shroud of anonymity. Some users even tend to like the low pressure environment of VR and do most of their social interaction through VR; Affective experiences in

VR are comparable to affective experiences in the real world. This is also indicative of trends throughout all of VR research and applications. Experiences tend to feel like real experiences due to the immersion and sense of presence individuals feel in VR, even despite lack of realism in some cases; Social VR is better suited for group interactions over one-on-one interaction. This is due to the lack of intimacy that is prevalent through VR. It is much easier to interact with other avatars and have fun experiences with them than to take a single avatar seriously, despite who is on the other end; Limited mobility, many of their participants wanted to take off their HMDs while still engaging with group members. This is a real problem with VR, which is why it is exciting to see new mobile technologies like the Vive Focus and the Vive wireless adapter; Difficulty in interpretation conversation protocol without nonverbal cues. Despite their use of animated mouths during speech, users still found it hard to tell when a user was going to speak due to the lag that is inherent with any network based application; Avatars tend to be representative of users. Most of their users tended to steer towards avatars that either looked like themselves or represented their personality in some way. Many users did try out an abundance of avatars, just to see what options they had, but some interestingly never swapped away from the avatars they chose at the very start of the experiment. It would be very interesting to see this experiment repeated with different populations of people and a follow-up study to see if users tended to continue with the relationships in VR or if they moved it to the phone or other online games and gaming platforms.

2.3 Software Engineering

2.3.1 Design Patterns

Design patterns themselves are repeatable solutions to common problems in software engineering. There are many design patterns, but there are three widely accepted categories for them. These design pattern categories are: Creational, Structural, and Behavioral. Each category has many different design patterns in each, so this work

will only reference a few of the most prevalent patterns that can be generalized.

2.3.1.1 Design Pattern Categories

Creational Design Patterns are about class, or object, instantiation. These patterns dictate how and when objects are created and destroyed. Factory based design patterns use an object to instantiate multiple different types of objects. Sometimes factories also handle the deletion of these objects when necessary. There are also builder design patterns that use inheritance to instantiate objects. Lastly, there is the singleton design pattern where only one of a singleton instance can exist at any time. This is done via the use of the static keyword found in most modern programming languages.

Structural Design Patterns are about object composition or how objects connect. The bridge design pattern separates an object's interface, which is how an object gets interacted with by other objects, and it's implementation, which is what the object actually does. The decorator design pattern dynamically adds functionality or responsibilities to an object as they are needed. The private, public, and protected keywords are also examples of structural design patterns due to the fact that they restrict, or open up, permissions that allow access and mutation to specific data or functionality.

Behavioral Design Patterns are about communication between objects. The command design pattern allows commands to be given by an object to another object. This can manifest itself in a typical master-slave architecture that is common in software engineering and computer science. There are also design patterns where commands get sent through multiple objects until the one with the responsibility gets activated, which is a bit different than the command design pattern, but still very similar. State based design patterns are also very common where, once a state changes, functionality of objects who are reading from that state will change.

2.3.1.2 Component Driven Design

Component driven design is the design pattern that is natural to incorporate in Unity. This design pattern is a structural design pattern, and similar to the composite design pattern. This design pattern has also been heavily incorporated into web development and general software engineering. Component driven design is very similar to the basic concepts of object oriented programming, where components are created to be reused and interact with each other. This is done on a larger scale in component driven design, however. Component driven design is the design pattern and design methodology where components are created to be parts of a larger whole. This larger design could be a game object, a web page, or a system. In Unity this represents itself as a few components attached to a game object that may or may not talk to each other. In software engineering, this can take many forms. For example, a tab based application can use a "Tab" component for each tab in an overarching tab list that manages events, ordering, and redirecting to specific pages.

2.3.2 Development Methodologies

Development methodologies in software engineering dictate how a software team works on a project and how the team communicates both inside and outside of the team. These methodologies don't apply just to software teams either, but that is the context that this thesis will be describing them in. Every methodology has it's strengths and weaknesses and these strengths and weaknesses can be product specific. It is also important to note that there are many different methodologies, but the largest ones that will be covered here are: Agile, Scrum, Extreme Programming (XP), Kanban, and Waterfall.

2.3.2.1 Agile

Agile development is based on incremental changes to the software requirements and iterating through them while requirements change. This set of methodologies involves the customer/end-user in these incremental changes by meeting with them regularly

and prioritizing what they decide is highest priority. This approach's goal is to make a working product at every stage, which means they always have something to show off to the end user to get feedback. The twelve principles behind the Agile development cycle are described in the Agile Manifesto [64]. It is important to note that many methodologies listed below are Agile, as Agile is a parent concept to many methodologies, not a methodology itself even though it often gets mistaken for one.

The advantages of Agile based development are quite vast, which is why it has taken off as a development mindset. The first advantage is that it provides a readiness to change. Due to the quick and versatile nature of Agile, it is easy to be ready and to make changes to requirements and pretty much anything else that is required. The second advantage is that developers generally end up with faster and higher quality deliverables. This is due to the granularity of the iterations in Agile development. Not only is it easier to manage bite size chunks of development, but it's also the methodology that most programmers think in to do their job correctly, so it comes very naturally to them. The third advantage is that everyone is heard. Team members are encouraged to communicate to get things done and everything is treated like a team effort. Customers are also understood in a more in-depth way, where they can choose priorities and play an active role in the development progress. It also means that the customers get the advantage of seeing frequent demos and can expect releases ahead of time. The last advantage that is going to be talked about here is the constant improvement of the project. Due to the fact that customers get more of an iterative role in the project, the project comes out to be a lot better and preferences are learned by the team much more quickly.

The disadvantages of Agile based methodologies come from it's very flexible and fast paced approach. Project plans are not very concrete due to the fact that they can always change. This makes it difficult to put down a delivery date. It makes it hard to put a ton of thought into the project as a whole, and instead, developers can only really think granularly and adapt it later. Time is also a huge issue with Agile methodologies. To make a good project with these approaches team members

not only have to be skilled and work in small teams, but they also need to have most, if not all, of their time dedicated to the project they are assigned to. It also means that developers generally have to be present for most/all of the project. Losing team members in these methodologies can be devastating, especially due to the next disadvantage. Documentation can be sparse or neglected. The fast paced nature of these methodologies causes documentation to fall through more often than not. New tools can reduce this downside, but it is still very prevalent. The last downside going to be talked about is the disparity between the vision for the project at the beginning and the end of development. Due to the constantly changing features and priorities that are prevalent with these methodologies, the difference in product can be huge and so it's hard for the software team to budget out both time and resources as there is no concrete plan in place.

2.3.2.2 Scrum

The Scrum methodology is an Agile process. Often considered the most popular approach to development. The Scrum methodology has fixed-length iterations, called sprints. These sprints last about a week or two. After each sprint, stakeholders, team members, and end-users will communicate about the next steps for the next sprint. This is where a "Product Backlog" gets formed. A "Product Backlog" is all of the features and requirements that the system has to have. Scrum have four common procedures for each sprint: Sprint planning, daily stand-up meetings, sprint demo, and sprint retrospective. Sprint Planning is where the highest priority, and likely next features, from the "Product Backlog" get shown to the team, and they decide which to put into the "Sprint Backlog". The "Sprint Backlog" defines what needs to be done in a given sprint. The "Sprint Backlog" is often a subset of the "Project Backlog", though the "Sprint Backlog" is generally more granular and development focused, while the "Project Backlog" is customer/user facing. There are also three roles in the Scrum methodology: The Product Owner, this is the end-user who has a vision for the product they want to have built and they try to convey that vision to the rest of

the team; The Scrum Master, this is often the person organizing meetings, dealing with challenges, and working with the Product Owner to ensure the current and next sprint go smoothly; and The Scrum Team, this is the rest of the software team, they plan for each sprint and decide how much they can get done for each sprint, each member of the team does not have distinct roles and instead get everything required done together.

The advantages of Scrum, firstly, mirror those of the Agile list above, as it is an Agile methodology. Scrum also has added transparency due to the daily stand-up meetings. These meetings show which member of the team is taking on certain issues, and gives each member their turn to speak and describe what they'll be doing for the day. This also allows the team to address any issues that might have occurred during the previous day's development. Scrum also boasts that it decreases project costs due to the constant communication, fast issue addressing, and continuous testing.

The disadvantages of Scrum also mirror those inherent with any Agile methodology. Lack of a concrete finishing date can make the scope of the project expand heavily. This also encourages stakeholders to expand functionality resulting in a project that doesn't get fully finished and is constantly being stretched further. Scrum also requires a large time commitment. Each member of the team is required to devote themselves to the project, and can't expand their usefulness to other projects. The team also has to set aside daily meeting time, on top of normal meetings that may be required throughout the team. Another disadvantage of Scrum is the fast turn around time between project definition and implementation. Most tasks can end up poorly defined, which increases the project's time and resources. Many corporations are starting to implement a "Technical Writing" position to help make this issue less common.

2.3.2.3 Extreme Programming

The first Extreme Programming (XP) project started in 1996. XP is an Agile methodology that further stresses communication with the customer. XP has much shorter

development cycles than Scrum's sprints, usually lasting about a week. XP focuses on continuous development, merging, and testing. This means that rather than merging all of the sprint's functions into the main line at the end of a sprint, XP requires that you integrate as often as possible, with working code, not necessarily at the end of a development cycle. The same goes with testing the code and application. The rest of the Agile methodologies covered can be applied to many different projects, not just specifically software development, and are just generic frameworks for project management and development. XP was built with software engineering specifically in mind. This methodology puts a large focus on programming practices and software engineering practices. XP encourages pair programming, collective code ownership, daily deployment, and test-first programming. This methodology also encourages incredibly fast builds, where the entire project should be built in under ten minutes total. XP also encourages teams to work directly with each other, often participating in paired programming and team conversations. Lastly, XP encourages a forty hour work week at most, so that the developers can stay energized and do good work without getting burnt out.

The strengths of XP lie in it's specific programming mindset and collaboration. Due to the fact that XP was designed with software engineering in mind, many developers find that this methodology is more natural feeling. It also encourages pair programming which can help make good code by sharing engineering knowledge. XP's even quicker development cycle and daily builds cause quick iterations and an always working build. XP also benefits from it's customer focused design, just like any Agile methodology, though customers can generally get a better feel for where XP projects are in their development due to the fact that these projects are almost always at the latest version. XP also helps out programmers due to the no overtime rule that's in place.

Weaknesses of XP lie in it's fast paced methodology and strict programming principles. XP is very opinionated with how engineers/programmers should build their applications with specific requirements in feedback times and the process of

development, release, and iterations. This can make it difficult for programmers to work effectively, due to being too focused on specific goals and procedures. It takes self-discipline to practice these specific procedures. XP also has an even higher time investment into the process of code generation and planning than other Agile methodologies due to the frequency of updates and communication with a pair programmer.

2.3.2.4 Kanban

Kanban is a Japanese visual framework. Kanban itself means "Visual sign" or "card" in Japanese. This framework uses a "Kanban Board" which has different swim lanes to put product features in. These swim lanes for software development are, generally: backlog, ready, coding, testing, approval, and done. These are the stages that each feature has to go through in order to be finished in Kanban. This used to be all that was involved with Kanban, moving features from one swim lane to the other as it happens, but now teams can measure and view progress and flow of the board. Software can help measure the effectiveness of the flow, cycle time, and measure the movement of features to determine efficiency.

The strengths of Kanban lie in it's extremely flexible and easy to understand nature. Kanban essentially just visualizes the Agile project and doesn't give much in terms of procedure, so it has all of the advantages that Agile has, except with more visualization. Due to it's visual nature, Kanban is incredibly easy to learn as, realistically, the only things that need to be learned are when to move a feature from one swim lane to the next, and what each feature, or color of a feature, means. The board also makes a more regular rhythm that teams can quickly get used to in order to keep or increase pace for deliveries.

The weaknesses of Kanban lie in it's lack of managerial abilities. Boards can be outdated and any feature that isn't currently being worked on will just stay on the board, left alone until it needs to be updated or someone clears it out. It is also easy to misuse the board and put either too granular, or too large of scope items on it. Kanban also suffers from a lack of timing. The swim lanes only give an approximation

for when items will be complete, not a hard deadline. This can be very frustrating for users/clients.

2.3.2.5 Waterfall

The Waterfall methodology is not an Agile based methodology. This methodology is strict and linear, rather than flexible and cyclical like in Agile development. Many Waterfall based projects are planned using a Gantt chart. A Gantt chart is a horizontal bar graph where the X-axis is time and the Y-axis is filled with parts of the project. Each bar indicates one part of the project and they stretch or shrink to indicate how long that aspect of the project will take. Each bar can start and end at any point in time, so that it accurately displays when and where a part of the project will start and end. The Waterfall methodology has eight rigid stages: Conception, this is where the project idea is created. It also assesses the project for positives, negatives, and initial budgeting; Initiation, this is where the team is hired or formed and you define objectives, scope, purpose, and deliverables; Requirement Gathering and Analysis, requirements are gathered and determined to be possible or not. This information is then put into a requirement specification document; Design, this is where design specifications are created from the requirement specification document and the overall design of the system is defined; Implementation, this is where the actual coding and implementation happens. The code is written from the design specifications in the previous stage; Testing, this is where code has already been completed and then is being more thoroughly tested; and Maintenance, this is where customers use the product and the team solves any problem that comes up. In traditional pure Waterfall models, each of these stages must be completed in turn and the team can not go backwards unless they're going to go through every stage again from the start. This doesn't work for programming due to the fact that testing and development must be done at the same time, over and over again, to make a functional and complete product. Teams, therefore, can often move back and forth through these two steps, violating the methodology. This is where the Iterative Waterfall methodology

comes in. This is a more modern approach to software development where a significant amount of planning and software requirements design is done at the start, like the traditional methodology, but the rest of the steps are completed per user story. This creates a much more modular, and much more cyclical, methodology that often results in stronger products.

The strengths of Waterfall lie in it's rigidity and planning. Due the rigid nature with planned development dates, Waterfall models are easy to manage. They're also the same pattern, so once you've managed or developed one project with the waterfall method, all of the other projects will have the same stages and have the same patterns. Waterfall also enforces progress due to the hard start dates and end dates. Waterfall also somewhat protects against missed deadlines due to the heavy planning process that happens before the work even begins. The last advantage of Waterfall is that it leaves behind a well documented paper trail for each phase. This is beneficial for stakeholders and managers, as well as for informing new members of the team where the project came from and where it's going.

The disadvantages of Waterfall lie in it's inability to change and non-iterative approach. Changes in functionality or requirements can't be easily achieved. Once the team completes a phase, they can't go back, so if the customer isn't happy with how a feature turned out and wants it done a different way, the team must start out from the beginning, which is time consuming and, therefore, expensive. Waterfall doesn't encourage communication between the users and the development team. Due to the fact that Waterfall models only deliver at the end of the project, the team can not obtain feedback from the stakeholders/users very often. Stakeholders/users also don't get to see the project until rather late in it's development cycle due to the project not mandating working code, like at the end of a sprint in Agile methodologies.

2.3.3 Computer Aided Software Engineering Tools

Computer Aided Software Engineering (CASE) tools are integral to modern software engineering. These tools allow a software engineer and a whole software engineering

team to get ahead of their competition. These tools have a vast variety of functions. To start, there are advanced Integrated Development Environments (IDEs). This software helps facilitate software development by integrating a source code editor, build automation tools, and a debugger. This alone is a CASE tool due to the automation of builds; however, these IDEs are getting more and more advanced with additional CASE tools and functionality. Advanced IDEs can allow multiple different programming languages, syntax highlighting, code completion, automated refactoring, integrated version control, multiple code searching automations, visual programming, multiple written language support, source control, plugin support, and many other features. Indeed, IDEs have a fast amount of features and are gaining new ones quickly. The largest new feature that can be seen in advanced IDEs is plugin support. This feature allows other developers and corporations to develop CASE tools that help facilitate their software and systems. This allows easier access to advanced systems, but with less overhead for dev-ops. For example, inside of the advanced IDE "Visual Studio Code" [65], any developer can download the "Angular Essentials" extension which allows Visual Studio Code to recognize and build Angular [30] apps. The same can be said for a Docker [67] extension or a Firebase [31] extension. It is really exciting to see approved extensions that are based on systems developed by the direct competitors of the IDE itself. This can be seen with the Angular and Firebase extensions, which are extensions to help support Google, who is a direct competitor of Visual Studio Code's developer Microsoft. While not direct collaboration, this truly improves many software engineer's working environment.

Another interesting case that is relevant to this thesis is Unity [104]. Unity is a CASE tool due to it's automatic attaching of scripts, and many other features. JetBrains's Rider is an IDE that is specifically designed for collaboration with Unity. This means that it is a CASE tool built for a CASE tool. This IDE helps Unity development further than a regular C# IDE due to it reading specific global names or settings that can be found in Unity, like a scene's name, and allows the developer to choose those names rather than having to guess and check at it. It also allows

debugging during Unity play, something that other IDEs do allow, and has a few unique features inside of Unity that are useful, like a direct code editor window in Unity. JetBrains [51] also has RiderFlow, which is a CASE tool built to overlay in the scene view and hierarchy for Unity. It allows saved camera placements, color coded hierarchy views, and saved objects to easily be moved to and selected at a moment's notice. These features help speed up the development process and provide better user experience to developers. There are also many CASE tools that Unity's package manager allows you to install from the asset store. These tools include better animators, terrain generation, automatic tiling, terrain editing, robotic integration and simulations, to-do lists, and so much more. Modular CASE tools, rather than integrated tools, is a much needed addition to almost any development environment.

The last kind of CASE tool integrate AI autocompleted code. One example of this is Tabnine [98] which can be added as an extension to many IDEs, including Rider. These kinds of tools have an AI that starts off knowing a programming language's grammar and then reads in all of the code in a developer's project to try and give accurate code suggestions to the style and expected outcome of anything the developer is writing. These suggestions are generally longer than default code suggestions found by default in some advanced IDEs. The suggestions received are generally full lines of code. While there are small cases where these tools have been useful in development, these tools are certainly not always accurate. Often, using the default code suggestions in IDEs, especially the ones geared towards the environment you're developing in, is more accurate and requires less editing after the fact than using this kind of tool. There is no doubt; however, that as the technology and AI improves these tools will be indispensable for software teams.

Chapter 3

A Virtual Environment For Fire Simulations

vFireVI: 3D Virtual Interface for vFire

This chapter first appeared as a Conference publication in ITNG 2020 [57].

Christopher Lewis, Ronn Siedrik Quijada, Frederick C. Harris, Jr. (2020) vFireVI: 3D Virtual Interface for vFire. In: Latifi, S. (eds) 17th International Conference on Information Technology–New Generations (ITNG 2020). Advances in Intelligent Systems and Computing, vol 1134. Chapter 41, pp 309-315. April 6-8, Las Vegas, NV. Springer, DOI: https://doi.org/10.1007/978-3-030-43020-7_41

Abstract

Wildfires cause severe amounts of damage to wildlife habitats and property. The most successful way of escaping safely or quelling a fire is to do so with communication, and as a group. While there are several other wildfire simulators, visualization and multi-user components are lacking or non- existent in most of them. vFireVI aims to provide a safe and accurate virtual environment for simulation and interaction with wildfires through multi-user collaboration, accurate terrain, and realistic fire spread. In order to do this, an interface was built between vFireVI and an earlier project, vFireLib, to allow transmission of simulation data back and forth. Combining these two allows for an intuitive user interface, responsive multiplayer, quick server communication,

and rapid simulations. All of this is done in virtual reality to provide a meaningful and immersive experience, where users can collaborate and test each other.

3.1 Introduction

Occupations that operate in dangerous conditions on a daily basis, such as military and emergency settings, face the issue of training employees for these dangerous conditions. Subjecting them to equally dangerous training exercises increases risk of injury, so other methods of training that provide a similar experience to on-site work are preferred, due to the minimal risk involved. In recent times, the growing popularity of consumer head mounted displays (HMD) has made Virtual Reality an increasingly viable option to provide minimal risk training.

In the case of forest fires, their immense scale and unpredictable nature make physical analogs costly and dangerous, thus having a virtual analog that has a minimal and flat cost is very efficient and useful. Using the virtual 3D environment provided by Unity, paired with an immersive HMD we created a safe and viable option for providing minimal risk fire safety training in optionally cooperative situations.

Virtual reality (VR) is fantastic at conveying a sense of immersion and user enjoyment; however, VR can also cause some users discomfort and detachment from reality. Through the use of Unity, VR is carefully implemented to provide a realistic 3D environment, and visualizer for the simulation. With minimal real world movement by the user, moving throughout the 3D environment reduces VR discomfort and motion sickness, which makes teleportation a good mode of locomotion for the 3D landscape, while still maintaining the user's autonomy while moving around.

The rest of this paper is structured as follows: Section 3.2 talks about the background behind this project, more specifically vFire and it's successors, and then related works, which are mostly fire simulators. Section 3.3 is where the design and implementation of vFireVI gets explained in detail, from the implementation of Unity to the Burning Simulation itself. Section 3.4 is Conclusion and Future work, which is where the paper shows its findings and explains future work that could come from

this project.

3.2 Background and Related Works

3.2.1 Fire Simulators

Another fire simulation was used for wildland two dimensional fire spread and was made by Finney at *et al.* [24]. This simulation approaches fire simulation through a unique perspective. All of it's data is based off of historical U.S. data. This includes weather patterns, wind speed, and moisture. It uses this data and then compares it against data from 91 fires occurring from 2007 to 2009. Their results consistently had fire sizes higher than the real fire, and consistently smaller farthest burn distance than the real fire.

H. Xue *et al.* [118] made a fire simulator that compares different combustion models in enclosed simulations. These combustion models are the volumetric heat source model, the eddy break-up model, and the presumed probability density function model in 3 situations; a room fire, a shopping mall fire, and a tunnel fire. Comparing each set of data, the authors in [118] found that none of the models are consistent over each situation. They suggest that there is a need for adequate turbulent combustion models.

H. S. He *et al.* [34] created a fire simulator that discusses the effects of fire on wildlife. The fire simulator incorporates fire, wind-throw, and harvest disturbance in terms of species-level fauna to determine patterns over large spatial and time domains. This simulator; however, does not predict individual events, it predicts the future of the species-level ecosystem.

3.2.2 vFire

vFire was a simulation developed by Hoang *et al.* [40, 41] in 2010. The simulation utilized a four-sided and six-sided CAVE™ virtual interface was used to display a reconstruction of the terrain near Kyle Canyon, Nevada. The simulation also used a

UI to simulate and modify various forest fire situations and terrain data. The height map and vegetation data, used in the simulation, can be seen visualized in Figure 3.1 and Figure 3.2, respectively. One of the largest issues with the application was that both the underlying simulation system and virtual interface were highly coupled, meaning that in order to update the fire model used, the entire application would need to be updated in order to use the new model. Another large issue with the application was that the virtual interface was created in OpenGL before OpenGL implemented pre-shaders, which caused huge calculations in any simulation, and made the project time consuming and difficult to upgrade or work with. In addition, if stakeholders would want to update the visuals, it would require a rebuild of the entire system. This would be true even if the underlying simulation had not changed.

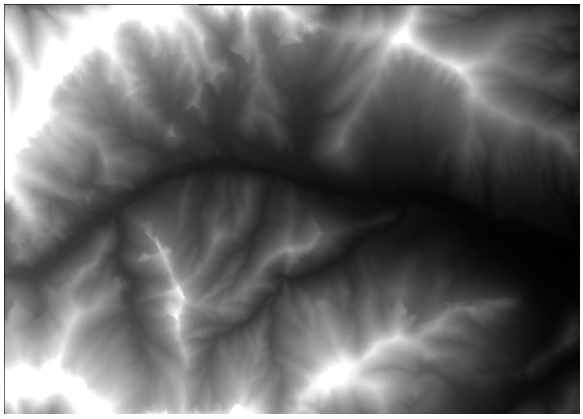


Figure 3.1: A height map generated using data retrieved from vFireLib

3.2.3 vFireLib

Due to the restrictions of the vFire simulation, a new system called vFireLib was created, in part by J. E. Smith [93], R. Wui *et al.* [117] , and R. Wui [115], to decouple the fire simulation simulation from the visual interface. vFireLib reinvents the original functionality of vFire as a RESTful interface, allowing the direct upload of fuel load, wind, and initial fire data to initialize the simulation. The simulation finishes by creating the resulting time of arrival map. This map provides data about what time each cell will set on fire and is returned from a REST call. This data

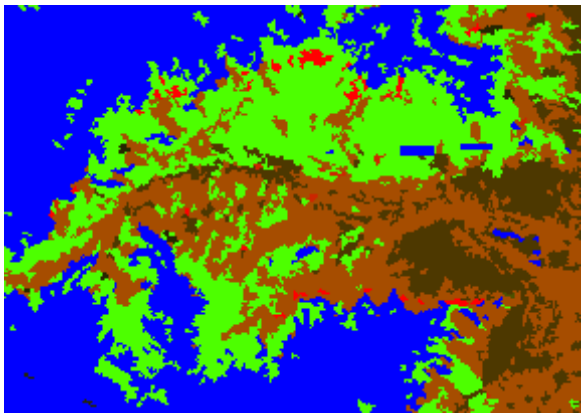


Figure 3.2: Vegetation map generated from the vFireLib fuel load index map; each color representing variable attributes.

can be used in any application. The project Harris *et al.* [39] developed a useful web interface, providing tools to modify and simulate various wildfires using modern browsers An example of the interface can be seen in Figure 3.3, which is a pixel map of the land the simulation is ran on. Two of the larger issues with vFireLib was that the simulation and web interface did not allow for fire spotting and the simulation was also rather slowed down by the addition of the interface.

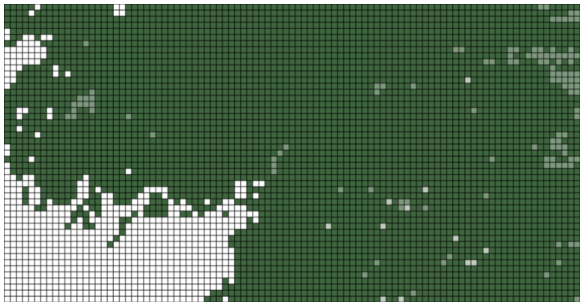


Figure 3.3: The visual interface, in VFireLib, corresponding to the area the simulation was ran

In order to rectify the issues of vFireLib, and provide the user using the web service or Unity application, the ability to modify and run fire simulations, vFireLib.v2 was created by Garcia [29] and influenced by [92]. vFireLib.v2 was rewritten to increase the speed of the sequential algorithm by implementing a parallel algorithm that was processed by the GPU. The simulation also included fire spotting computation.

Implementing spotting required computing where an ember would emerge once a fire had reached a hot enough intensity, and where that ember might land considering wind, gravity, moisture, and fuel data to determine if a new flame would ignite.

3.2.4 VR Simulators

VR Simulators can be found in many different forms. Much of the research in this area is in surgical applications. A unique example of this is by J. M. Albani *et al* [2], which details the work of VR assisted robotic surgery simulations throughout the years. It also incorporates one of the current commercial surgical system, the da Vinci™robot. The article also describes possible future applications of the simulations and the remaining challenges that need to be fixed.

Another VR simulator that fits into the field of surgery can be found in an article by A. G. Gallagher *et al.* [27], which describes the times in which a VR simulation is actually helpful in teaching and the reasoning on what makes it useful. It describes the use of simulations for minimally invasive surgery only, but it could be used as a training tool for other types of surgeries as well. Their conclusion and results specifically make the case that VR simulators are only impactful when integrated into an already good education or training program that involves actual technical skills.

A fire simulator in virtual reality also exists, it was created by M. Cha *et al* [13]. This simulator uses computational fluid dynamics to calculate certain quantities: toxic gases, heat, smoke, and flames. The paper's focus is on creating a training simulation, so that civilians, members of the military, and new firefighters can experience wildfires second-hand. Overall, the study provides a clean framework for accurately calculating quantities of fires, and inputs into a simulation. There isn't any spreading of the flames and the paper discusses using this framework for building sized situations, like evacuating, so that the fire not accurately spreading, isn't a large deal.

3.3 Implementation

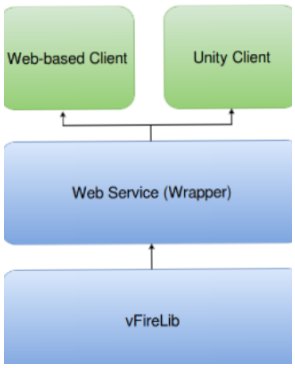


Figure 3.4: The connection between projects developed for vFireLib

As seen in Figure 3.4 there are many components to the overarching simulation of the modern vFire. vFireLib is the starting point, where the actual simulation gets done. The next step, Web Service, is the REST interface that exists to communicate and run vFireLib. From the Web Service, two clients are split off that communicate with the REST interface found within the Web Service. The first being the Web-based Client. This client exists online to communicate with the server. The other being the Unity Client. This client is vFireVI. The reason this client exists is to create a visualization of the simulation that is user friendly and can act as a realistic environment to help bolster the avoidance and preparedness against wildfires.

Due to how vFireLib fire simulation discretizes its simulation data to a grid, the REST interface provides a 2D array of several components of the fire simulation, ranging from wind data to vegetation type and density. vFireVI takes advantage of the highly parallel nature of this forest fire simulation data by offloading most of the work to the GPU through Unity's ShaderLab language and OpenCL Compute Shaders.

3.3.1 Unity

Unity is then used to show that data in a 3D environment by stepping through the burn chart, obtained through the Rest Interface, which is obtained by the server

running the simulation with new data. Next, the simulation must spawns the players and ticks through time, incrementing through the burn chart until nothing is left. Once the burn chart is empty, the simulation ends. Unity is also used to implement the 3D environment in virtual reality.

Mirror, a plugin for Unity, is used to implement a multiplayer aspect to the simulation. The multiplayer aspect of the simulation was implemented through the use of a client-host model. This is where the first one connected to a multiplayer session acts as a server or host for the rest of the players/clients and is a client themselves. If the simulation has to be changed at all, the host receives the information from the REST interface and updates the host's terrain. Once the terrain has finished updating for the host, that terrain is sent to all of the clients the host is connected to. Once this is done, the simulation continues. All stepping through the burn map is done from the side of the client, so the terrain wouldn't have to be sent every server tick. The movement of clients gets sent to the host, the host then updates its models, and sends the updated locations back to all of the clients this creates some latency, but not enough to hinder the simulation in any meaningful way. All of this project can be visualized as the Unity Client side of Figure 3.4 from R. Wui *et al.* [117] which references the connection between this project, vFireLib [93, 115, 117], and vFireLib2 [29].

3.3.2 In Game

vFireVI utilizes 2 different user roles. The first being the user role, "Runner". A Runner, as the name implies, runs around the environment and can jump as their only movement and interactive options. The second user role is the "Overseer". An Overseer can move more quickly than a Runner, and can fly as their modes of transportation. Another feature of the Overseer is that any Runner can't see the Overseer. This is to allow for an environment where someone can watch a user, from another room, without the user knowing about it, which could be useful in research. The locomotion and UI for the players were carefully decided through examination

and review of multiple research papers, such as M. Nabiyouni *et al.* [71] and M.M. Davis *et al.* [20]. It was settled to use a radial menu and for locomotion, the motion of pulling yourself along the terrain for the Overseer and Teleportation for the runner as they provide, generally, the best mix between possibility of motion sickness and usability.

As for the situation deployment, the program starts with a low opacity 3D interface and basic terrain in the background. This user interface has options that allow the user to join multiplayer, change aspects of the fire simulator, and change the terrain through various tabs, windows, and input fields. After the user completes setup and starts the simulation, the custom data selected by the user gets sent to the REST server to be processed by the server and then the burn map is returned back. Then, Unity runs scripts to load the terrain and the burn line from the available data. Both types of players, Overseers and Runners, spawn in after they load into the simulation and the 3D environment has been built.

3.3.3 REST Data Parsing and Terrain Generation

The REST interface reinvented by vFireLib provides plain text files containing data of the simulation via an array of integers; however, a 2048 by 2048 cell simulation has over 4 million entries to parse. To speed the parsing stage up, a compute shader is used to split the data into lines and parse each row into the buffer that will contain the final array of integers. From there, we can pass the final data back to the CPU or transform the array into a texture that can be used in rendering the 3D environment. This texture is then passed to the Terrain Generator.

Terrain generation is primarily handled using Unity's built-in terrain system, which provides functionality for setting height data, trees, and various other details. For the height of the terrain, the parsed data is transformed into a 16 bit integer array like in Figure 3.1, which is then passed to the terrain system. While there isn't a clear way to directly replace the height map contained in the terrain system, Unity provides functions to replace the height map with an array of integers via C# scripts.

For the terrain material, Unity uses what is called a splat map, which is a texture that defines what ground material is used at which point in the terrain using the RGBA channels to represent the strength of each ground material. A custom shader was written to take advantage of this existing functionality by using the RGBA channels to describe generic variable attributes of the terrain. An example of the splat map can be seen in Figure 3.2, where blue controls unburnable areas, green controls the fuel density of the area, and red controls the fuel moisture.

3.3.4 Forest Generation

For forest generation, the project uses a 16 bit integer map generated from the previously parsed data. The data holds indexes that represent a terrain type. Terrain types are defined by the data obtained from U.S. official land surveys. The indexes in the data are compared to a user defined dictionary that pulls the fuel load density and tree type, and then spawns an instanced tree at the location on the terrain relative to where it was sampled on the index map. This system allows for variable forest types, allowing it to simulate a wider range of flora.

Forest generation requires different types of vegetation *e.g.* shrubs, trees, and grass. Thus, a great deal of importance is placed on allowing for these different types, and having example figures for each. vFireVI has a model for each type of tree specified from U.S. official land surveys. This means that any new vegetation type can't be read into the simulation until a new model is created and tied to the specific representation of the vegetation data.

3.3.5 Burning Simulation

The visual burning of the forest is handled through a set of custom shaders built on top of the existing Unity shaders that effect the rendering for all visual elements in the scene. For all the custom shaders, a map called time-of-arrival, which contains data pertaining to the exact time each cell sets on fire, is referenced in order to determine the visual state of the fire simulation. A sigmoid function is used on the

sampled texture to convert the float value of an area into a value that represents its current burning state given the current simulation time. This function is then used to determine the visual aspects of the area.

For trees, the burn state function is used to visually set the tree on fire, and fade away small branches and leaves based on its current burn state. To support varying burn times of different forest types, a secondary map is used to store the burn color and burn time for each tree, assuming trees don't migrate. The code for the rendering is split into two different shaders handling the tree trunk and leaves respectively.

For terrain, the burn state function is used to visually show the fire line at a given time, represented by a glowing line. The thickness of the line is determined by the speed at which the area sets on fire. The color of the fire line changes based on the type of fuel being burned, as well as other user defined traits. The terrain texture will change based on the burn progress as well, where burned areas change from their original material to a burned rubble material.

Figure 3.5: The burn map generated from a successful run of vFireLib's simulation. float values are passed in and saved into an exr format to preserve the 32bit floating point time of arrival information.

3.3.6 Example Simulation

An example data set was given from the aforementioned vFireLib simulator so that this project could continue where it left off. This data set is of Kyle Canyon, Nevada as seen in Figure 3.6. Along with the location data we were given terrain height data

as seen in Figure 3.1, local vegetation data as seen in Figure 3.2, and the vFireLib simulator itself had moisture and wind data contained within it.

Figure 3.6: A satellite view of the simulated area in Kyle Canyon, pulled from Google Maps.

vFireVI ran the simulator and took in the data received from the simulator to create the 3D representation of Figure 3.5. This 3d representation is shown in Figure 3.7. The strong bright yellow line represents the current burn line. On the left side of this figure there are representations of already burnt out trees, on the right there are representations of the not yet burned trees. There are also places on the left side with not burnt trees. This is due to there being terrain data that represents areas that can't be burnt overlapping with vegetation data that shows vegetation in that area. This is due to poor data rather than imperfections in the simulation.

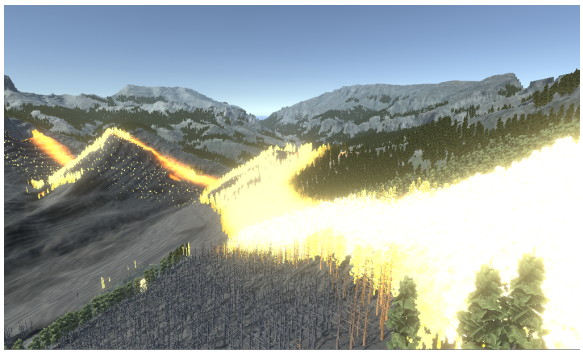


Figure 3.7: The fire line of the simulated area in Kyle Canyon, visualized by the Unity simulation.

3.4 Conclusion & Future Work

3.4.1 Conclusion

vFireVI is a virtual reality based visualization environment of a fire simulation. The simulation can be communicated through a REST interface server from a previous project called vFireLib which allows for quick changes to the environment and rapid simulations. Virtual reality makes this simulation, and others like it, viable options for training of all sorts, at all levels. The physical analog provides little safety, especially compared to the minimal risk of the head mounted displays. vFireVI also allows for responsive multi-user simulations. The virtual reality element tied in to the multi-user simulation allows for collaborative training which makes it even more effective.

3.4.2 Future Work

Dynamic Collection of Data The way vFireVI gets data currently is using presets that already have correct data stored within it. These data sets could be scraped from a terrain database website and then formatted to allow for vFireLib to parse the data.

Fire Textures vFireVI allows for dynamic texture swapping; however, there are no accurate textures available for fire or smoke representation within the simulation. The current textures are just seen as glowing and then the terrain textures are swapped from “not burnt” to “burnt”.

Role Functions Currently, vFireVI has two roles, the “Overseer” and the “Runner”, these roles only really change the movement modes of the user. These roles could be expanded upon to allow for meaningful differences between the two. There could also be an increased game element to the simulations which might cause the user to be more invested in the simulation.

Better Multi-User Unity recently removed support for multiplayer games on their platform. vFireVI was made using a Unity plugin called “Mirror”. This causes the

multiplayer to be a little slow and it is peer to peer. It would be better, in some cases, to create a server to peer system to allow for less lag and less issues with server or peer updates. It would also be better to use a multiplayer system that is native to the game engine environment.

Editing Data vFireVI uses data given by vFireLib to run. There are two ways this could be improved. The first is letting a user edit the data directly in the Unity user interface. Another is to allow the user to edit data live during the simulation, perhaps as the "Overseer" role. Data that could be edited includes, fire start point, additional fire points, vegetation, rain fall, and wind. This would provide a good way to allow friendly user editing of the simulation without having to look at the raw ASCII files.

3.5 Acknowledgment

We would like to thank Rui Wu, Connor Scully-Allison, and Andy Garcia for their help in operating their past implementations of the web interface for vFire, and providing us with their simulation and simulation data.

This material is based upon work supported in part by the National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Chapter 4

Eye Tracking in 360°VR Video

An Application for Simulating Patient Handoff Using 360 Video and Eye Tracking in Virtual Reality

This chapter first appeared as a Conference publication in CATA 2022 [56].

Christopher Lewis, Sven Diaz-Juarez, Steven J. Anbro, Alison J. Szarko, Ramona A. Houmanfar, Laura H. Crosswell, Michelle J Rebaleati, Luka A Starmer, and Frederick C Harris, Jr. (2020) An Application for Simulating Patient Handoff Using 360 Video and Eye Tracking in Virtual Reality. In: Bidyut Gupta, Ajay Bandi and Mohammad Hossain (eds) 36th Conference on Computers and Their Applications (CATA 2022). EPIC Series in Computing, vol 82. pp 141-149. March 21-23, Online. EPIC, DOI: https://doi.org/10.29007/82j6.

Abstract

Virtual reality (VR) is a relatively new and rapidly growing field which is becoming accessible by the larger research community as well as being commercially available for entertainment. Relatively cheap and commercially available head mounted displays (HMDs) are the largest reason for this increase in availability. This work uses Unity and an HMD to create a VR environment to display a 360°video of a pre-recorded patient handoff between a nurse and doctor. The VR environment went through different designs while in development. This works discusses each stage of it's design and the unique challenges we encountered during development. This work

also discusses the implementation of the user study and the visualization of collected eye tracking data.

Keywords: Medical Simulation, VR, Eye Tracking, 360 Video, HMD

4.1 Introduction

This work describes an application used for displaying a 360°video in VR to students practicing to become doctors and nurses. The 360°video started with a doctor coming into a patient room while a nurse is taking care of a patient, in this case a training mannequin. The doctor and nurse initiate a patient handoff while the user watches. A patient handoff is the very common action of transferring patient care from one medical professional to another. The doctor and nurse get cut short as the patient has a crisis that requires both of their attention. During this crisis, they quiz the user on what they discussed during the patient handoff, where the user has to respond verbally. This scenario was given to the user two times. Once at the beginning of their training for patient handoff, and once at the end of their training. During this scenario, this application records eye tracking data, which is collected from the HTC Vive [46] HMD with TOBII [101] beta eye tracking integration. This hardware is no longer in beta, and is an early implementation of the eye tracking that is a part of the HTC Vive Pro Eye HMD. The virtual space itself was created using the Unity [104] game engine and then integrating it with SteamVR and the TOBII beta eye tracking SDK, which is the beta version of the SRanipal Unity SDK[44].

The rest of this paper is structured as follows: In Section 4.2 we discuss a few related works to this project focused on 360°video, eye tracking, and VR medical simulators. In Section 4.3 we cover the implementation of the user study and eye tracking data visualization. In Section 4.4 we cover different versions of this project's virtual space with specific notes about corresponding eye tracking with objects/people in 360°video. In Section 4.5 we discuss the overall conclusions found from this application. Finally, in Section 4.6 we introduce future work that would make this application better and allow further research in this area.

4.2 Background and Related Work

There are quite a few umbrella terms that might classify this work, from software engineering to human-computer interaction. The unique aspects of this work, however, come from an intersection of three main topics: 360°video displayed in VR, eye tracking in VR, and VR medical simulators.

4.2.1 360°Video in VR

360°video displaying in VR is a somewhat new use case for VR. Due to the lightweight portability that HMDs provide, VR is seeing much more prevalent use in research, industry, and in general entertainment. This increase of use means that streaming 360°video, rather than having the user store video on their machine, is a much needed addition to make VR more dynamic. However, streaming 360°video in high resolution is not an easy task due to the fact that these videos contain much more data than a standard video stream. Much research has gone into fixing this problem, but more recent research uses neural networks, and other machine learning techniques, to predict where a user might fixate next inside of a 360°video[23]. This technique is very useful in limiting the total amount of information that needs to be rendered for any given 360°video, but it requires a significant amount of user data on a wide diversity of videos to train. This problem is currently being investigated, but a useful step in the right direction is the creation of datasets containing user eye tracking data on specific 360°videos[59]. This data set captures both sensor data and content data. Sensor data are pieces of data collected from the HMD sensors, like head position and orientation. Content data are pieces of data collected from the video, like saliency maps and motion maps.

Perhaps one of the largest innovations for 360°video streaming in VR is within the advent of tiling. Tiling is the process of breaking up specific sections, or tiles, of a 360°video and streaming only tiles that are relevant, or close to relevant, to the user. This means only the tiles of the video that are being currently viewed by the user,

or about to be viewed by the user, are streamed to the user. Two problems exist within this approach, one lies within the ability to know what the user might want to view next, and the other problem is in how small the tiles are in each approach. An approach that aims to fix both of these problems is within an optimization framework for cellular networks[81].

4.2.2 Eye Tracking in VR

Eye tracking in VR is yet another new use case for VR. With the integration of eye tracking built directly into commercially available, though expensive, HMDs [45], researchers have more access to eye tracking technologies for VR than ever before. With more researchers getting their hands on this new technology, an introduction to this field is very prudent. The authors of [16] have done just that. Their work shows valuable insights into the creation of eye tracking applications for VR and how to deal with certain inherent aspects of VR, like motion sickness and fatigue. They also ran a pilot study for an example application. All of this culminates in a very helpful introduction into research for this field.

Another application to integrate eye tracking into VR is in facial expression classification[38]. This work uses a convolutional neural network (CNN) to tie a user's expression to a perceived emotion, through the use of images collected from the HMD's eye tracking cameras. Interestingly, this work shows that using an emotionless image from each user serves as a good baseline for the CNN. Overall, they found their approach to be about 70% accurate, a great step towards realizing facial expressions into VR for greater immersion in multi-user applications.

4.2.3 VR Medical Simulators

VR medical simulators are vast in terms of their diversity. Some simulators focus on specific surgeries or type of surgery, like ArthroSim[102], a VR knee and shoulder arthroscopy simulator. Some simulators are general purpose, like ORamaVR [77], a general purpose VR surgical training simulator. These simulators are vast and often

times very specific, but researchers have found that the visual realism, interaction realism, and tactile realism are lacking in most trainers [87]. The interaction and tactile realism aspects of this problem can be clearly shown with the technology used for each simulator, more specifically, if the simulator uses a commercial HMD or not.

HMDs are general purpose and very good at specific types of interaction; however, they do not excel at certain aspects of what surgical simulators require, which are dexterity and feedback. Most HMDs use rather bulky controllers, due to the need for greater accuracy inside of VR. This is an issue for surgical training due to the decrease in dexterity the controllers provide. These controllers are also very different in shape, size, feel, and weight than any tool a surgeon might need to use, causing a disconnect from what they are trying to train for. The controllers themselves also only have haptic feedback, which can't give meaningful feedback to the user. Haptic feedback does not provide the proper resistance or feel that surgeons can expect to work with, making a further disconnect from interaction realism and tactile realism. Lastly, visual realism often suffers due to a lack of diversity in the development team for these applications. Most researchers are not proficient in 3D modeling, so visual realism suffers in a large amount of these simulators, something 360°video and techniques like photogrammetry aim to improve.

4.3 Implementation

4.3.1 User Study

The user study for this application consisted of male and female students practicing to become nurses and doctors. Each user used the application twice. The first time was before they received training on patient handoffs and the second time was after they received training. During both rounds, each user was ushered into a standard patient room where they would sit down in a chair and be given pre-written instructions, as seen in Figure 4.1.

Once the instructions were read, the user put on the HMD with the help of

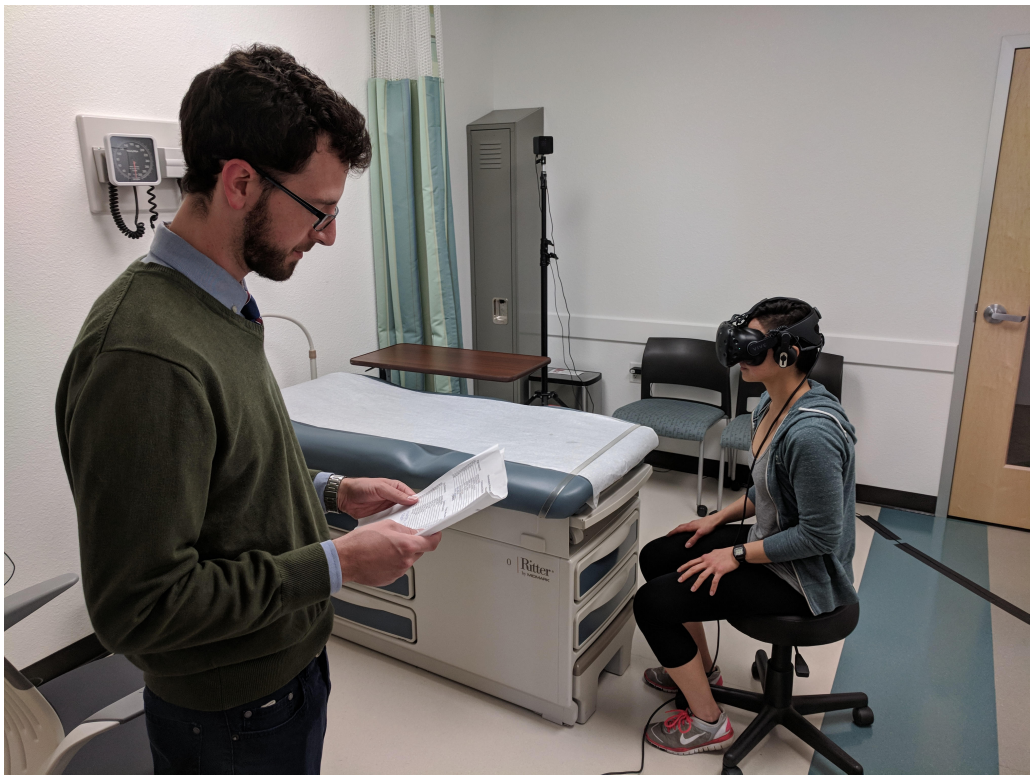


Figure 4.1: A picture taken of a participant while being given pre-written instructions.

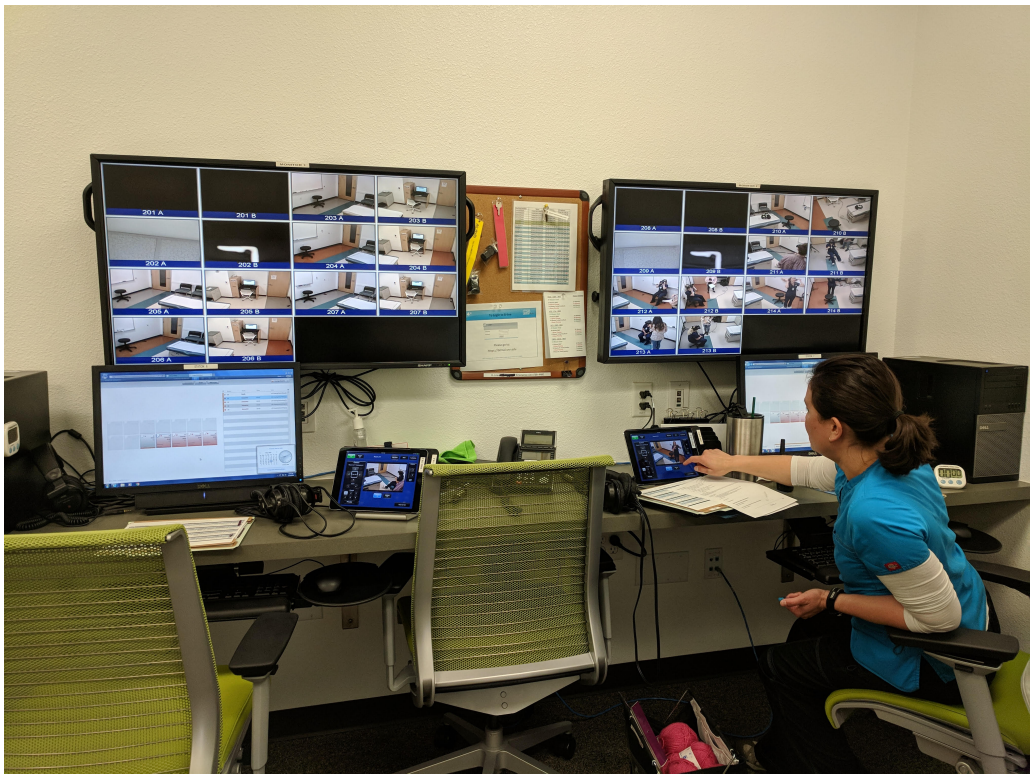


Figure 4.2: A picture of the standardized patient room control station.

the authors and followed the instructions. The instructions described the process of calibrating eye tracking and how the user should interact with the application. The user was required to watch the 360°video and to respond to what was said in the video verbally. During this time, the person giving the instructions stayed in the room to address any concerns the user might have and to assist the user in getting out of the HMD. The user was also recorded and observed during their entire participation inside of the standardized patient rooms, as seen in Figure 4.2. For more information on this user study, and the findings from it, please see Anbro *et al.*[5].

4.3.2 Eye Tracking Data Visualization

Eye tracking data was gathered using a beta TOBII SRanipal SDK for Unity. The data collected was the participant ID (the ID is just the order in which the users participated), a boolean stating if the user is a nursing student or a medical student, and a list of timestamps indicating when the user looked at a given area of interest

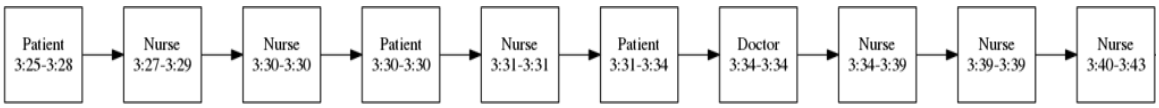


Figure 4.3: A small section of visualized eye tracking data notating when a given user looked at specific AOIs and for how long.

(AOI). This data was then evaluated for the time of first fixation for each AOI and how long their attention stayed on the AOI. This occurred throughout the scenario - before, during, and after there was a "crisis" with the patient/mannequin. The data was also evaluated for total number of fixations and total time fixated throughout the video on each AOI.

All of the data evaluated was then combined into a timeline for each AOI fixation, as seen in Figure 4.3. This figure shows each time a user fixated on a different AOI. It is important to note that each element of this figure is of a new fixation for each AOI and when in the video it happened. When an element is followed by another element of the same name with a gap in time between them, it notates either a drop in fixation, or that the user was fixating on something that was not labeled as an AOI. When an element shares it's fixation time with another element, it indicates that both AOIs were being fixated on, meaning they shared space with one another during the video. The reason we allowed AOIs to conflict in timestamp is due to the fact that it is not clear which AOI the user is trying to fixate on at a time. For example, a user might be focusing on the patient, but during the "crisis" event the nurse and doctor share the same space as where the patient's AOI would be, so the user would end up focusing on their AOIs instead. The user in this situation is still fixating on the patient, or at least that was their goal, but they are now also fixating on the doctor or nurse. Including all timestamps for all AOIs solves this issue of conflicting AOIs and it gives researchers using this data for their evaluation a better grasp on what happened at any given data point.

4.4 Experimental Setup

The virtual room needed to accomplish two main goals. The first was to play 360°video without any defects or distortions. The second was to allow eye tracking measurements to be taken at any point in the video feed. Different iterations of the virtual room were then developed to accomplish these goals, and correct weaknesses in previous iterations.

4.4.1 Skybox

The skybox was the first iteration of the virtual room. A skybox is a default screen for most game development platforms, and since this project used Unity[104] for it's development platform a skybox was easy to create. A skybox is a rendering of a 3D pseudo-infinite space. This space is similar in look to a sky that goes on forever with no ground under it. This skybox surrounds the user constantly, even after moving. At this point in the development process, the authors didn't have the beta eye tracker from TOBII[101] or the SRanipal Unity SDK[44] (which was still in development throughout this study), but did have 360°video to render, so we used the skybox to render the video instead of the infinite space. This displayed the video accurately and with no noticeable defects or distortions. This was done by continuously updating the skybox with the next frame in the video. However, once the authors received the eye tracker and integrated the eye tracking API into the application, we realised the skybox wouldn't work as the display medium due to how the eye tracking API works.

4.4.2 Eye Tracking

The TOBII eye tracking API took measurements from an updating eye tracker on the inside of the Vive HMD [46]. These measurements would be either the angle towards the object the user is focusing on from the user's virtual HMD or a null measurement, meaning either nobody is in the HMD to look or the user is not fixating on anything. This angle measurement would then need to be raycast from the middle of the virtual

headset in the direction of the angle returned by the HMD. This raycast would then need to collide with an object that is looking for raycasts to hit it, which then calls a customizable script. This script was used to write the time the object was hit into a file for each participant, and was later used for data analysis. Due to the fact that skyboxes do not interact with collisions from raycasts, the skybox would not work with eye tracking.

4.4.3 Box

The second iteration of the virtual room was a hollow box. This box is an object in the unity space that can interact with the raycasts at each of it's sides. The user's avatar was put in the center of the box and a video renderer was added to it's inner sides. The same script to render the 360°video onto the skybox was used on the regular box, as a skybox is similar to a box with a video renderer on the inside. This made the 360°video render smoothly, with no defects or distortions, and allowed the eye tracker to work on the 360°video. But, the box itself is a single collider, this means that the raycasts sent from the eye tracker will always result in the box being hit. A method was needed to correlate what the user is looking at in the video. This is where AOIs come in.

Areas of interest (AOI) are locations inside of a video that are relevant to the audience of a video. AOIs are generally objects of some relevance or individual people. For this application the relevant AOIs were the doctor, nurse, patient, monitor, and table as seen in Figure 4.4. These AOIs were identified by the authors, and then we created invisible boxes that would move if the correlating AOI in the video moved. This means that when the nurse moved, so did her corresponding AOI.

The way we made the AOIs move was by manually shifting the coordinates of the AOI in the Unity world space everytime the AOI shifted in the video. We recorded the coordinates, and the correlating time in the video, to then have the shifts be repeated automatically during the simulation. These manual shifts mean that the AOI measurements are very close approximations to the ones in the video, but not

Figure 4.4: A screenshot of the 360°video playing for each participant

perfect.

The authors also noticed an error in the design of the box as a display medium for eye tracking. Any eye tracking coordinates obtained from the raycasting on the background, the box itself, would be incorrect. This is due to the recording format of the 360°video being a sphere.

4.4.4 Sphere

The final iteration of the virtual space was the sphere. The sphere was very similar to the box iteration, but it solved the issue of displaying a sphere inside of a box. This issue is created in the corners of the box, where the coordinates when raycast to from the center of the box would be deeper than they would be in a sphere, making them inaccurate.

4.5 Conclusions

This work created an application to be used in patient handoff training that integrated 360°video within VR. This application also mapped 360°video with a user's eye tracking data in real time to be used in evaluating training methods for the pa-

tient handoff. The application is novel due to the intersection of three techniques; VR, 360°video rendering, and eye tracking. The application itself was built using a beta version of the TOBII SRanipal SDK for eye tracking. This SDK, and the TOBII eye tracking hardware, are now integrated natively with the HTC VIVE Pro Eye HMD. This work also provides details as to what worked, and did not work, within the VR space to allow for eye tracking to be used effectively.

4.6 Future Work

There are several things that can be looked at to improve work in this area. This section presents several of the areas we think would be beneficial for future researchers to consider.

AOI Tracking: AOIs were created and moved manually. The AOIs were paired with a list of locations and timestamps for this specific video. Using computer vision that paired AOIs to their associated person or object for any video would limit the manual labor needed to recreate this application with a different video.

Heat Map: A heat map for where a user was looking would be a great addition to this project. The heat map would allow another visual metric to be displayed and evaluated by researchers. This heat map could be static and evaluate overall fixation by the user, but it could also show the fixation over time for any given video during a playback. Using these heat maps and comparing them to other user heat maps can show a large amount of metrics and show aggregate data useful in research and marketing.

Virtual Environment: A virtual environment for patient handoffs would be a good change for this project. Allowing the user to move around and interact with things inside of this virtual environment would allow for deeper immersion and greater use of VR technologies. This change could allow many different scenarios to play out,

not just patient handoff. It also opens up the possibility of having users interact in a multiplayer and collaborative environment which might further their knowledge and experience with skills like the patient handoff.

Streaming Video: To take full advantage of the VR format for medical training, it would prove useful to stream video rather than host it on a user's local machine. With AOI tracking established, it is feasible to create many 360°videos that allow eye tracking data to be collected. To allow this broader goal to be reached, a new UI for selecting videos/trainings would need to be established and the list for selecting these videos/trainings would need to be updated from a server with new videos/trainings when they get released. Much research has been done on improving the large resource load that streaming 360°video requires [23, 81], some of this research must be implemented to allow for streaming video for medical training to be feasible in a broader scope.

4.7 Acknowledgements

We would like to acknowledge the UNR School of Medicine and the Orvis School of Nursing for their collaboration and for having their students participate.

This material is based in part upon work supported by the National Science Foundation under grant numbers 2019609 and IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Chapter 5

Multiplayer in VR

Virtual Reality Multiplayer Interaction and Medical Patient Handoff Training and Assessment

This chapter first appeared as a Conference publication in CATA 2022 [56].

Christopher Lewis, Daniel Enriquez, Lucas Calabrese, Yifan Zhang, Steven Anbro, Ramona Houmanfar, Laura Croswell, Michelle Rebaleati, Luka Starmer, and Frederick C Harris, Jr. (2022) Virtual Reality Multiplayer Interaction and Medical Patient Handoff Training and Assessment. In: Bidyut Gupta, Ajay Bandi and Mohammad Hossain (eds) 36th Conference on Computers and Their Applications (CATA 2022). EPIC Series in Computing, vol 82. pp 141-149. March 21-23, Online. EPIC, DOI: https://doi.org/10.29007/82j6.

Abstract

Virtual worlds have the potential to mirror many aspects of real life. Immersive virtual worlds constructed through the use of Virtual Reality (VR) are useful in simulating the technology, equipment, and practices of many different fields. In the medical field, VR can be heavily relied upon to circumvent a wide variety of tools, human resources, and other objects that may be limited or difficult to procure at any given time. As a result, the goal of this research was to develop a low-stakes virtual environment (VE) in which medical students could practice developing skills necessary to their profession. As such, this environment needed to mirror, as closely as possible, an environment the medical students would see frequently during their

practice. The result of this work is an application for use in patient handoffs, a situation where patient care is transferred from one medical professional to another. In order to achieve this, this work created a multiplayer VR environment with an immersive virtual world simulating standardized patient rooms and standard mediums of communication and interaction between users. While doing so, a framework was developed as the need for VR multiplayer, VR with voice communication, and a VR interaction system was seen to be needed for future VR multiplayer applications. This framework can be used to construct more applications for communication fueled environments, like the patient handoff.

Keywords: Virtual Reality, Multiplayer, Framework, Software, Simulation, Eye Tracking, Photogrammetry

5.1 Introduction

The medical field, and those who work in it, can be considered the backbone of a modern society. This is evident in normal life, but overly obvious in a global pandemic. A task often taking place in any hospital around the world is the patient handoff. The patient handoff is the transfer of patient care from one medical professional to another. Patient handoffs happen at a very high frequency, often occurring multiple times per patient, per day. It is estimated that the most serious adverse events in hospitals are caused by communication failures, mainly to do with patient handoff errors[35]. These errors can cause very large problems to both patients and medical professionals. Through virtual reality, this work introduces a tool to streamline and standardize patient handoff training, a step forward in addressing these errors.

VR is rising in popularity throughout the world in many different sectors. Major sectors innovating with virtual reality are the education and medical sectors. In fact, there has been major growth in the use and innovation of VR for training medical professionals and for healthcare in general. VR provides these sectors, and many other sectors, with engaging and immersive educational scenarios that help reinforce taught behaviors in a kinesthetic way. These kinesthetic learning methods help to

engage the user, which cements the lesson into their memories by providing multiple learning styles for the user to learn from [48].

5.2 Background and Related Work

Patient handoffs happen very regularly across any hospital, making the handoff one of the most pivotal and important actions medical personnel should learn to do effectively. Yet, handoffs have the potential to cause a large amount of harm due to their somewhat regular missteps and miscommunications [53]. This highlights the need for greater care and plentiful innovation in the education and training of medical professionals, particularly in the skill of patient handoff. The SBAR is an effective innovation in the communication and standardization of handoffs. It has been shown that the SBAR has had an incredibly positive impact in the areas it affects [70].

With the improvement in the structure of the handoff that SBAR creates, innovations have also taken place around the education or implementation of handoffs. One such innovation is with the user study this application was built for[6].

5.2.1 Medical Training Simulators

Innovations in medical training can have wide reaching impacts for the overall medical community. One such innovation is within the use of haptics for medical training simulators. Coles, Meglan, and John in [18] cover the use of haptics for training in simulators through various scenarios, such as: palpation, needle insertion, laparoscopy, endoscopy, and arthoscopy. Overall, the authors found that surgical simulations incorporating haptic feedback provides a richer training experience, compared to the simulations that don't.

VR based medical training simulators have also come to fruition. One such medical training simulator is for periodontal training [60]. The authors put haptic feedback into a VR simulator to allow the user to fully experience working on teeth. This is ultimately a safer approach since it doesn't involve a patient or real teeth. The authors also found, through various studies and expert interviews, that their work

provided a realism that could serve as a, "useful instruction tool with high teaching potential" [60].

5.2.2 VR in Medicine

VR is seeing more prevalent use in the medical industry as an alternative for training applications for surgeries. New technologies, such as the simulations made by ArthroSim [102] and ORamaVR [77], are some of the most innovative VR surgical simulators.

ArthroSim [102] is a VR knee and shoulder arthroscopy simulator. ArthroSim uses a dedicated machine for high precision surgery training. It uses a dual-monitor display that reflects the training simulator task and data, as well as the simulated imagery of a camera used for these surgeries. This machine focuses on precision due to its dedicated machinery and its use of haptic feedback in training. AnthroSim also focuses on realism, where the precise use of haptic feedback would be similar to physically performing surgery on a person. The simulated images of the camera also use accurate anatomy to reflect where the user is currently working. This technology provides a much safer and efficient solution to training surgeons, opposed to letting surgeons learn on mannequins or on patients.

OramaVR [77] is a general purpose VR medical training simulator with multiplayer capabilities. One of the largest benefits of using this system is its reliance on mass-produced VR head mounted displays (HMD). Not only are these HMDs much cheaper than a dedicated machine, but they can generally be used interchangeably with other varieties of HMDs. This allows the application to be used by a large variety of users and to be flexible for new technology and innovations within VR technology.

5.2.3 Multiplayer VR Simulators

Multiplayer VR is a relatively new technology. Although some applications, mostly video games, use multiplayer VR as a form of entertainment, not much research has been done surrounding it and its intrinsic problems. One such intrinsic problem with

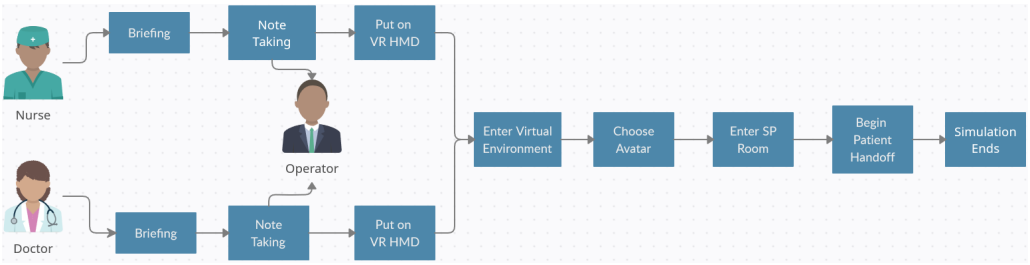


Figure 5.1: The execution order of the application in use during the user study.

VR that has been studied is room-scale multiplayer VR. Sra in [96] addresses the use of a Galvanic Vestibular Stimulation system to allow the user to avoid collisions with their environment while not breaking the user's immersion. This technology looks incredibly promising for the future of VR for commercial use, as not every person has a large room to navigate in.

Multiplayer VR can also have very interesting and useful applications. One such application is in the world of construction. Du, Shi, Mei, Quarles, and Yan in [21] detail the use of multiplayer VR for building walkthroughs. A building walkthrough is a tour of a building, often before it is built, through the use of simulations, models, and CGI images. Building walkthroughs using multiplayer VR allowed this project's stakeholders to view a 3D virtual model of a building that they otherwise couldn't experience aside from inside a less interactive medium. It is also important to note that this application also allowed the users to verbally interact with each other. Due to the immersive nature of VR applications, the users of this program would also be able to experience the building in the exact same scale as their avatars, causing a more robust and informative walkthrough.

5.3 Implementation

5.3.1 Process of Execution

As described in Figure 5.1, before the program is started two participants were briefed separately as to the current condition of a unique simulated patient. Each participant wrote down any information they chose about the patient to help convey the

Figure 5.2: A participant configuring their avatar in the starting room.

information needed during the patient handoff. Each of these files are given to the operators of the program to be uploaded into the program.

Once the participants have been properly worn the HMD, the simulation will start. Both participants are then put into the VR world together. They each start in the same room as seen in Figure 5.2. This room allows the participants to choose their avatar and appearance freely. Once either participant is ready, they can point and press their remotes at a sign that will teleport them into the standardized patient room as seen in Figure 5.3.

When both participants had teleported to the standardized patient room, they could start the patient handoff for each patient. Each participant had the notes they made about the patient displayed on their left hands in the form of a clipboard. This was done so that they could glance at it and help themselves complete their patient handoff using the written information, like in a physical working environment.

Finally, when both participants completed their patient handoffs, the simulation ended and they took off their HMDs.

Figure 5.3: The standardized patient room put into VR via photogrammetry.

5.3.2 Eye Tracking

For this work the Vive Pro Eye was used for eye tracking through the interface of HTC's SRanipal SDK [44]. The observer and the recording could observe timestamps and location data of when and where the user was looking throughout the application. This data could be used to determine where in their handoff the users were engaging with the other user, looking at the patient as seen in Figure 5.4, or getting distracted by the environment.

5.3.3 Recording

To monitor user experience in this application, this work recorded the movements, communications, and interactions using RockVR video capture [85]. This video capturing is recording the exact screen that the observer views, including eye tracking lines, and audio from both participants. The physical room during the user study [6] was also being recorded using an in-room camera that connected to an overseer station as seen in Figure 5.5. Lastly, some participants were recorded in a more close-up manner, as seen in Figure 5.6.

Figure 5.4: The standardized patient room put from the perspective of the observer with eye tracking visible.

Figure 5.5: The overseer station used in the user study.

Figure 5.6: A participant engaged in the user study with an operator in the background.

5.4 Room Creation

The VR patient room was a very accurate approximation of a standardized patient room. This room was created to have the participants in an immersive space that is, very likely, familiar to them. To generate this room, this project needed to do three tasks accurately. First, the appearance of the room needed to emulate a standard patient room. Second, the room had to feel approximately the same size as it would in real life. Third, the appearance of the objects, the equipment and the dummy needed to appear as similar to real life as possible, with training equipment mirroring standard tools that are used in medical practice. Both the first and second tasks were done using a technique called photogrammetry, while the last task needed to be done, largely, manually.

5.4.1 Photogrammetry

Photogrammetry, broadly, is the technique of finding the correct size of virtual objects that correlates with the size and spacing of the physical objects they are associated with. The entire room, and the full body mannequin, are both generated using

photogrammetry for 3D modeling. These models were exceedingly large in terms of the storage space they required on the computers. These models were so large, in fact, that Unity [104] struggled to render them. Thus, we had to lower the polygon count of the objects. To do this, every wall was replaced with larger flat polygons, rather than a large amount of small polygons. Other parts of the room were then cleaned to remove gaps and other wrong textures, all while replacing these features with larger polygons that would take up less rendering time. This removes some of the finer resolution of this technique, but ultimately doesn't change the appearance of the room to any significant degree.

5.4.2 Room Objects

A few of the room objects were not available to have photogrammetry done to them. This includes the computer monitor, the cart of medical equipment, and the IV bag and trolley. All of these room objects were instead created using computer-generated imagery (CGI). Each of these elements had to be placed and sized directly emulating the room's photogrammetry. This created a small amount of scaling issues, as the objects couldn't be exactly the correct scale as we had no real life object to compare to the virtual room. All room objects can be seen in Figure 5.7.

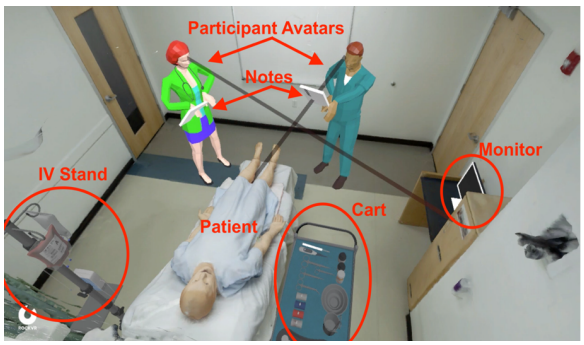


Figure 5.7: The room objects that were made using CGI and the patient mannequin.

5.5 Interaction

5.5.1 Avatars

The participants had the ability to change the avatar of their own personal character. This allows them to change their own virtual appearance to their liking to either fit their role or fit their own personal character. The participant can change to either wear a medical smock or to wear scrubs to fit their role of a doctor or a nurse. These outfits had a variety of different options associated with them as seen in Figure 5.2. They could also change their skin color, gender, and hair color accordingly to whatever they deemed fit.

5.5.2 Voice and Audio

The participants are required to communicate verbally to each other in order to perform the patient handoff. This includes both producing audio and receiving audio throughout the application. This was accomplished using the built-in microphone and speakers found in the HMD. As a result, this application included the ability to communicate with one another via voice chat. Due to this voice chat integration, both participants can execute the patient handoff in almost exactly the same ways they would in the real world.

5.5.3 Clipboard

Each of the participants were briefed before entering the virtual world on a particular scenario of a patient's current status. Each of the participants were then instructed to write down everything they wished to convey during the patient handoff. Theses notes were then transferred into the application and displayed on a clipboard in their left hand to emulate a typical medical setting. These notes were only available to be seen by the person who wrote them.

5.6 Framework

5.6.1 Interactables

In many instances of multiplayer VR, some form of visible interaction can occur between objects and/or users that all users should be able to see. In this framework, interactable objects in the shape of spheres were made that could be picked up and thrown that the other player would be able to see move and interact with the environment. If a player were to pick up the object by moving to the object, putting their controller into the object, then pressing the trigger, the object would then mirror the movements of their controller. Upon release of the trigger, the object would be thrown in the direction and speed that mimics the direction and speed of the controller.

These types of interactions require data transfers to be broadcast between all of the users involved in that instance of VR. This is to ensure the concurrency and accuracy of these interactions on all user instances. To accomplish this in the multiplayer framework, a client to server system was implemented. A client to server system has a server that communicates between the players and gets the inputs necessary for program functions such as movement and interaction updates and it shares it to all users in that instance of VR. The server also stores important information such as synchronized objects and the active scene. As a result of this server model, if a user were to grab an object, the object would exist on the server's independent storage of the object. This means that if the user were to move the object, the user needs to communicate with the server that they have grabbed the object and have moved it. The server would then broadcast to the user holding the object and all other users that the object is being moved. This handshake is very slow, and the movement of the object would appear choppy and inaccurate. However, to counteract this problem, our framework has the ownership of the object swap from the server to the player in contact with the object. This means the server is no longer keeping track of the object, but is getting frequent updates by the user holding the object as to the position, rotation, and velocity of the object. This means that there would be no latency

in moving the object for the user holding the object. When the interactable is then released by the user, the server takes control of it again, causing the updates to only be broadcast from the server and thus lowering latency between the users.

It is important to note that while the user study only allowed two users into the VE at any given time, this framework allows for many users to collaborate. The varying factors for precisely how many users are: the number of assets used in the game, the internet speed of the users and server, the power of the user's computer (largely GPU based), and the number of assets set up for networking/interaction.

5.6.2 Voice

Voice communications across users are synchronized to the other player. This means that if one player speaks the other player is able to hear the audio. In this work, the medium of voice communication was done through Photon PUN Voice chat [110], with the microphones being connected to the HMD as well as the audio from the other person coming through the speakers of the HMD. Voices were also being played and recorded on the observer's computer.

5.7 Conclusions

VR is seeing more implementation in training and education than ever before, particularly inside the field of medicine. With medicine being one of the most important parts of civilization, it is critical to improve the training of common sources of error and miscommunications, like the patient handoff. The work presented is an application where the standardization of patient handoff training could be effective in achieving reduced errors and reduced miscommunication.

This work demonstrated a framework for user communication and user interaction in multiuser VR environments. It also created an application specific to typical medical environments and training in a particular medical scenario. This application was tailored to a standardized patient room and attempted to mirror common interactions and points of communication for evaluation of communication such as

voice recording, video recording, and eye tracking. As a result of this application, a framework was developed to streamline the process of these multiplayer interactions that could allow for future development for VR multiplayer interaction applications.

5.8 Future Work

5.8.1 VR Cross-compatibility

As a result of this application being made to be compatible with the HTC Vive Pro Eye's eye tracking software [108], the application doesn't work with other HMDs. This means that the program is not compatible with other popular VR hardware, such as: Oculus Rift/Quest [75], Vive Wave [109], mobile VR, etc. A future iteration on this work could allow for the inclusion of different mediums of VR. The removal of a hardware requirement would allow accessibility to people who own a different VR headset. This would, however, disable eye tracking for the HMDs that couldn't support it.

5.8.2 Avatar Eyeball Movement

As a result of eye tracking being a prominent feature of this application, it could be beneficial to share the eye movement of the other participant. A situation in which this could be useful is to experiment with the level of eye contact occurring in the hand off. Eye contact is crucial for flawless communication to occur, so allowing other participants to see what another participant is looking at could be very impactful.

5.8.3 Avatar Lip Tracking

For communication to occur smoothly in VR, increased immersion is very important. The more immersion the participants experience, the more seriously they would take the simulation, which would cause communication to become more accurate. If a participant is able to see another participant's mouth moving, it might increase the immersion of that participant. An example of technology that could be used is to

implement Vive's facial tracking technology and synchronize facial expressions to the other player over the network [107].

5.8.4 Object Interaction

In medical environments, it could be useful to have the medical equipment synchronized to other players. This would allow a user to show an application of specific equipment on an object to other users, how to use a stethoscope for example. It would also be more realistic to see objects, that other people are using, move and interact similar to the real world. This may improve the immersion of the users which may improve the effectiveness of the training.

5.8.5 Notepad Improvements

The virtual notepad that is being held by the participant may see some utility to share notes if these notes were synchronized across different users. This means that it may be useful for participants to share their notes/objectives to the other users. Allowing the user to dynamically create their own notes is also an important consideration, as creating your own notes is an important function for effective communication. Current note taking technology in VR is limited, as it is difficult to precisely write things down without a physical pen-shaped object to write with. This is an issue with the precision and maneuverability of the standard VR controllers. It is also difficult to type in VR due to the fact that the user is generally standing up or moving around the environment. Standard controllers have issues with precision and speed for typing in VR on a virtual keyboard.

Acknowledgements

We would like to acknowledge the UNR School of Medicine and the Orvis School of Nursing for their collaboration and for having their students participate.

This material is based upon work supported by the National Science Foundation under grant numbers 2019609 and IIA-1301726. Any opinions, findings, and conclu-

sions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Chapter 6

Typing in VR

Typing in VR Using Voice Interfaces

This chapter is the basis for a publication which is in the process of being submitted.

Abstract

Virtual Reality (VR) has many advantages compared to traditional computer interfaces; however, there are a sizable number of deficits that VR needs to solve to be more widely adopted. Arguably, the largest of these deficits is typing within VR. This work aims to shed some insight into successful typing methods for VR through the use of a user study and a comparison of input methods. It was found that a combination of dictation and a 3D input method lead to better results than solely dictation. It was also found that testing input methods with multiple types of input culminate in more varied and detailed results.

6.1 Introduction

Typing is critical in any modern computer interface. Typing is also a daily occurrence for most humans on the planet today. We type in both work and leisure environments. For some people, it is their primary form of communication. For something as critical as this, it is quite shocking that modern VR applications, generally, have little to no typing involved. When typing is involved, it is generally a slow process, especially

when compared to a modern keyboard. Typing in VR often involves individually selecting letters on a virtual keyboard interface. This is a much slower process than simply typing using modern keyboards.

This work aims to speed up the typing process in VR using modern approaches and provide a methodology for reviewing input methods inside of VR. The largest addition in this project, compared to modern keyboards, is in using dictation as a primary form of input. Two input methods are created in this text. The first is using an edited form of dictation. This edited form allows the user to input any letter or, generally, any character found on a modern keyboard, as well as write whole words and sentences at a time. The second input method is a combination of an edited "drum-like keyboard" [11] and dictation. This edited form of the "drum-like keyboard" displays special characters first and allows the user to swap to an alphabetized keyboard. The reason for starting the "drum-like keyboard" with special characters selected is due to the expected use case for the keyboard is to enhance dictation, not to type full sentences. The keyboard should be used to spell a single word that is hard to pronounce or hard for dictation to process. The authors created this distinction to, hopefully, facilitate understanding of when and why users might want to use the alphabetized keyboard opposed to spelling out words or acronyms using dictation.

The user study associated with this work tested for: words per minute (WPM); characters per minute (CPM); and errors per minute (EPM). The user study was broken up into four main typing challenges. These typing challenges are: URLs, Email addresses, sentences, and paragraphs. The reason these were chosen is due to the fact that they encompass almost everything the modern user of a computer might type regularly. The user study implemented a pre-test survey which collected demographic and experience data, and a post-test survey which collected general feedback and ideas for improvement. Lastly, the user study issued the System Usability Scale (SUS) for each input method, to test what the users thought about the usability of the input methods.

The rest of this paper is structured as follows: Section 6.2 describes a general background for typing in VR with various input methods and user studies; Section 6.3 details the implementation of the typing methods, requirements for the application, and the organization of the experiment/user study; Section 6.4 details the results obtained from the user study including WPM, EPM, SUS scores, and user responses from the post-survey; Section 6.5 discusses the results found in Section 6.4; Section 6.6 finalizes what information this work has found and gives the opinion that the way we currently measure typing speed needs to be expanded on; and Section 6.7 details expansions to this work that would further research, allow researchers to obtain better data, or would generally allow this work to expand.

6.2 Background and Review of Literature

6.2.1 Input Methods

The classic example of VR typing methods is the "point and select" or "raycasting" method. This is a method of selecting keys on a virtual keyboard with a VR controller. The VR controller in this example sends out a raycast to the keyboard and displays itself to the user. The user would then move the controller so that the raycast hovers over a specific key and then they would press the select button to have that key input into the system. One application that uses this method is Google Earth VR [32].

Another example of VR typing methods is the "punch typing" method [119]. This is a method of selecting keys on a 3D keyboard with a VR controller. This method accomplishes typing using a collision based system. The user takes their controller, or another object that can facilitate collisions, and virtually hit the keys. The collision between the keys and the object/controller causes the key to be pressed. Keys in this method can be arrayed into a myriad of positions and can often be manually moved by the users.

The "split keyboard" or "two-thumb touchpad" [94] method is an input method for typing in VR with both controllers at the same time. Most other input methods

could use both controllers at the same time, but would generally be quite uncomfortable for the user or they couldn't, generally, be used effectively. This input method aims to make use of both controllers by allowing each controller to operate distinct cursors on the virtual keyboard. This method is somewhat comparable to the "point and select" method in that they both use virtual keyboards and an, almost, cursor. However, this method is much more user friendly and much more precise. The "point and select" method is susceptible to shakes, jitters, and tremors from the user. This makes every character the user inputs a bit more of a struggle than just moving over the cursor like in the "two-thumb touchpad" method. The "point and select" method also doesn't work as well with both controllers being used to select keys as both hands would need to be extended, causing more shakes and tremors.

The "drum-like keyboard" [11] input method is almost a combination of the "point and select" method, the "punch typing" method, and the "two-thumb touchpad" method. The "drum-like keyboard" uses the VR controllers and extends a sort of drumstick (a stick with a ball on the end) out from the tips of the controllers. The 3D keyboard is displayed in front of the user where the user can then hit the keys with the drumstick, causing the key to be pressed. This allows the user to press keys with both controllers at the same time and allows for relatively quick input. The keys on the keyboard are usually at slightly different Y-coordinates based off of their row, with rows towards the user being lower than rows in the back. The keyboard itself is generally rotated towards the user as well, to allow every key to be seen at any time.

There are also a variety of input methods that do not involve the use of controllers at all. The first of these input methods is called, "dwell typing" [33]. This typing method, while not exclusive to VR, has been integrated into VR by using the middle of the HMD as the cursor that allows the user to wait/"dwell" over any key on a 2D keyboard to select that key. These approaches tend to be designed to help those with disabilities in allowing them to type without any additional and/or expensive equipment.

"Gaze typing" allows the user to focus their gaze on any key in a 2D keyboard

and dwell until the key is selected. "Gaze typing" is an evolution of "dwell typing". The most difficult part in designing these dwell based input methods is in deciding how long the user is required to dwell before making a selection. Make the dwell time too long and the WPM will go down, make the dwell time too short and the user can select incorrect keys on accident. There is quite a bit of research into what a correct dwell time is, but most research finds that it is related to the experience a user has with these systems [63]. Another interesting approach in this area is within the use of detecting the next most likely key to be pushed, and decreasing it's specific dwell time while increasing other key dwell times [68].

6.2.2 User Studies and Comparisons

VR research is steadily increasing it's use of user studies and comparison studies. As VR is becoming more of a main stay in both commercial and public use, the research surrounding its use is moving away from pure implementation details and moving towards user studies that test unique and helpful features.

A very relevant comparison study to this work is [10]. Which is a comparison study between four controller-based VR text-input techniques. These four input techniques are: Raycasting; drum-like keyboard; head-directed input; and split keyboard. Raycasting is the technique most commonly used in VR applications for text input and is analogous to the previously discussed "Point and Select" method. It involves raycasting from the tip of the VR controllers to hover over a 2D floating keyboard. The user would, most often, press the trigger button to input whatever key is being hovered over by the raycast. The drum-like keyboard is the same technique that this work is using, except it's primary focus is on alphabetical keys in a QWERTY fashion. Head-directed input uses the rotation of the head to select specific keys and is analogous to the previously discussed "dwell typing" method. The cursor is in the middle of the user's field of view and the user would hover over the key to select it. Split keyboard input uses both controller's thumb pads to move around a cursor for each controller in a 2D keyboard that is split in half. This method is analogous to

the previously discussed "two-thumb touchpad" input technique. This work found the mean WPM for each input method did not exceed twenty-one, with the drum-like keyboard being the fastest min and max WPM range. Due to this paper's findings, this work is using the drum-like keyboard as a part of the tested input methods.

One publication that details a user study that tests a unique and helpful feature is [80]. This work details a system for continuous identification and authentication of users in VR systems. This system uses a variety of body relation and movements to allow for a somewhat accurate identification system in VR based off of very simple tasks in VR. These tasks include: pointing, grabbing, walking, and typing. It trains a random forest classifier to create the identification process. They found an accuracy for each individual task and reached a highest accuracy rate of about 40%, but it would be interesting to see what the identification accuracy rate would be if they combined all stages and all tasks for an overall combined accuracy. As headsets and other VR capture devices get more accurate and allow for further granularity of motion capture, this could be a very accurate way to authenticate users.

6.3 Implementation

6.3.1 Software Engineering

The identified functional requirements for this project are found in Table 6.1. The identified non-functional requirements are found in Table 6.2.

6.3.2 Technology

This work uses Unity [104] as it's main development platform. We also used an HTC Vive Pro Eye [108] as the main HMD. Unity's built-in "DictationRecognizer" was used for basic dictation. Unity's "DictationRecognizer" is built on top of the Microsoft speech recognizer [99]. The reason the authors didn't use Microsoft Azure's Cognitive Speech Services SDK, despite Microsoft explaining it is generally better, is due to the financial trade-off for slightly better results. It was also identified that the free Unity

Table 6.1: The identified functional requirements

FR #	Functional Requirement Description	Priority
FR01	User can type any character found on a traditional keyboard	1
FR02	User can dictate any character found on a traditional keyboard	1
FR03	User can delete any character	1
FR04	Key presses should be clearly shown	1
FR05	Any input should be output to the same UI	1
FR06	Any input should be checked for validity	1
FR07	When a user finishes a stage the next stage should appear	1
FR08	The system will seamlessly switch between stages	1
FR09	When a new stage appears all input and sample text should be replaced	1
FR10	User can select and move to the testing scene	1
FR11	User can select and move to the main scene	1
FR12	User performance data should automatically be collected for each stage	1
FR13	User performance data should automatically output for each stage	1
FR14	When dictation is used the dictation hypothesis should be displayed	2
FR15	User can use dictation hypothesis as input text	2
FR16	Incorrect typing or dictation should be clearly shown	2

Table 6.2: The identified Non-Functional Requirements

NFR #	Non-Functional Requirement Description	Priority
NFR01	User can use the controllers to collide with and type a key	1
NFR02	The system shall be well documented	1
NFR03	The system will have sample text to be used in each stage	1
NFR04	Keys should move down and then back up when pressed	1
NFR05	All input text will be managed by an intermediary writer	1
NFR06	The intermediary writer will put all input into the same UI	1
NFR07	The intermediary writer will validate input text against the sample text	1
NFR08	The intermediary writer will transmit to the system if input text is incorrect	1
NFR09	The system will support user interface interaction using the point and click method	1
NFR10	The system will use select-able buttons to move the user between scenes	1
NFR11	User performance data will be written to a file titled the start time for the program	1
NFR12	Every user interaction and dictation will be written to a full log with timestamps	2
NFR13	User can press a button to use dictation hypothesis as input text	2
NFR14	Input text and keys should turn red if an incorrect character has been entered	2

"DictationRecognizer" would work for our purposes. The authors also used Unity's XR Interaction toolkit, version 2.0.0, for basic VR setup and OpenXR support.

6.3.3 Dictation

Dictation using Unity's "DictationRecognizer" required quite a few additions/changes to their algorithm in order for it to be used for our purposes. The goal for the dictation input method was to allow any text to be input at a user's discretion. The way

this dictation recognizer works is by generating a preliminary "DictationHypothesis" and then finalizing it into a "DictationResult" when the user is done talking. The "DictationResult" had a few complications, however.

The "DictationResult" doesn't have contextualized output in most cases. The result would allow the user to say contractions (I've, Haven't, Wasn't, etc.), which shows contextualized output of special characters. For most special characters; however, the dictation algorithm would either not allow the special character to be placed or the special character would be placed, but the word would not. For example, if the user were to try to dictate "I care, period." by saying, "I care comma period period", the "DictationResult" would output as, "I care,..". This shows that the "DictationResult" changes what was verbally said into a special character. This example also shows that there would be no way to dictate the example sentence, as any instance of the word "period" would change into the special character. This works in the same way for commas. In contrast to this, the "DictationResult" would not accept other special characters in the same way. For example, if the user were to try to dictate "#Hashtag" by saying, "hashtag hashtag" or "sharp hashtag", or any other naming variation, the "DictationResult" would output exactly what the user said and not replace anything with a special character.

The "DictationResult" has another problem. Single characters can't be input for things like acronyms, to spell out a particularly difficult word to pronounce, or a word that might not be a part of the dictation dictionary. For example, if the user were to try to dictate "ABC" the "DictationResult" would output, "hey be see". Thus, not only does the "DictationResult" not take into account single letters, but it places spaces in between words by default.

To correct these issues, we created something called a "command phrase". This "command phrase" consists of a "CommandWord" followed by a "statement". This "CommandWord" is a word that dictation can recognize and is set at run-time by the user. The "CommandWord" for the user study was set to the word "command" for all users in order for the study to take less of the user's time. The "Command-

Word" can be said before any character to dictate an intended "statement", which is just a character the user wishes to input. The "statement" can be any letter in the English alphabet or any special character found on a modern QWERTY layout keyboard. This implementation means that the "DictationResult" needed to be modified to remove the automatic swapping of periods and commas with their special character. For example of both changes, if the user were to try and dictate "I care, period.", the user would need to dictate, "I care "CommandWord" comma period "CommandWord" period". The "command phrase" in this example sentence is both ""CommandWord" comma" and ""CommandWord" period". This example also shows that at any point in dictation, the user can input a "command phrase" along with the rest of the dictation. This same approach works with individual letters and other special characters. For cases with multiple synonyms, for example the special character "#", which can be called a sharp, pound, or hashtag, can be called by any of those synonyms and dictation will produce the special character when led by the "CommandWord". These synonyms and wording for each letter are put into a text file that is read in at the start of the program that equates the special character with the synonym. While this approach does complicate typing, it allows a VR user to type anything they wish, albeit with a bit of practice. The largest issue with this approach is if the "DictationResult" does not output a known synonym or wording for any character. For example, if the "DictationResult" output "sea" instead of "see" for the letter "c", and we didn't include that as a possible synonym, then the model would not understand and just output, " "CommandWord" sea" rather than replacing the text with "c".

To correct the automatic spacing after each word, we implemented a variant of the "command phrase". The variance comes from the "statement" portion of the "command phrase", where instead of identifying a character a user would issue a "general command". A "general command" is a word that is associated with a specific function of the input or dictation system. For example, we created a "general command" using the word "spaces". When the user issues this "general command",

the dictation system no longer inserts automatic spaces between words or at the end of the "DictationResult". There are a large amount of "general command"s that could be created, but we have created only what was necessary to allow the user to use dictation for any input they'd traditionally be able to use. We have created "general command"s that allow the user to backspace, insert a space, toggle capitalization of individual letters, toggle automatic spacing, and issue a full-backspace.

6.3.4 Input Methods

As seen in Figure 6.1, the basic setup for input methods is quite simple. Any input method that we want to include in a comparison study or a general user study sends any input it receives into an input manager. This input manager itself has three jobs. The first is to capture any and all input received and to update the user's input text accordingly. This includes backspaces. The second job is to allow the authors to implement rather unique features that you won't find by default on a modern physical keyboard. The example for this function is the whole word deletion function, which we call a "full backspace". This is an input that we created in both input methods to allow the user to delete a full word. This is particularly useful with voice dictation

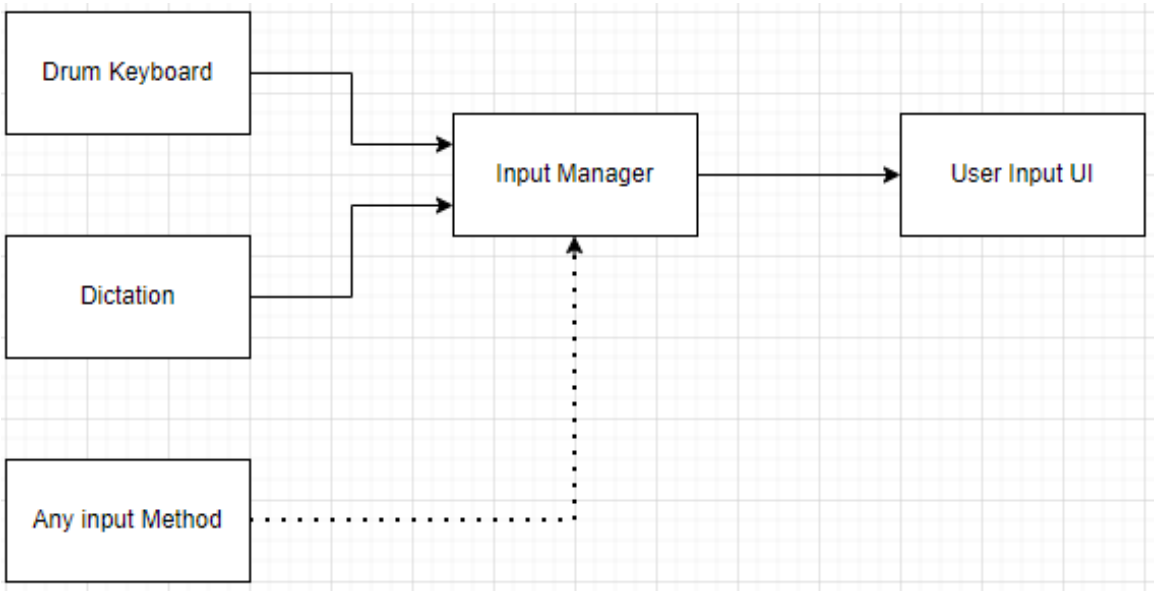


Figure 6.1: A block diagram detailing the basics for input method setup.

as it can sometimes detect completely incorrect words. The third job of the input manager is to detect if the input is correct or not and then send out appropriate commands to other objects. This is used in the instance where the user makes a mistake when typing, or corrects a mistake when typing. If the user makes a mistake, the input text and their virtual keyboard (if they are using a keyboard) turn red. If the user goes back and corrects their mistake, the input text and their keyboard turn back to their normal color.

The edited "drum-like keyboard" (drum keyboard) displays special characters by default, but it is designed to be fully adaptable to any key set. A key set is the set containing the characters to be displayed, and used as input, for each key in order. When the application starts, the "key manager" sets every key in the drum keyboard to match a specific key set. The key manager can also be told at run-time to change the active key set. This feature is used to swap between special characters in a custom format, lowercase QWERTY-layout characters, and uppercase QWERTY-layout characters, though it could be used just as easily with keyboards for other languages besides English, as well as different keyboard layouts like Dvorak and Colemak.

6.3.5 Experiment Organization

The experimental organization falls into four different stages. One stage for each input type. Each stage is run through twice, once with each input method. These stages are picked at random to allow the data gathered to not be influenced by the familiarity of the input techniques and allows a less biased look at the data.

The application also has three distinct scenes for the experiment. The first scene that users see is the "Menu Scene". This scene gives the user the choice to select between the remaining two scenes in a user interface. The mechanism for interacting with this user interface is similar to the "point and click" or "raycasting" input method described earlier in this work, except not used on a 2D keyboard. Once selected, the application will take the user into that scene. This scene can be viewed in Figure 6.2.

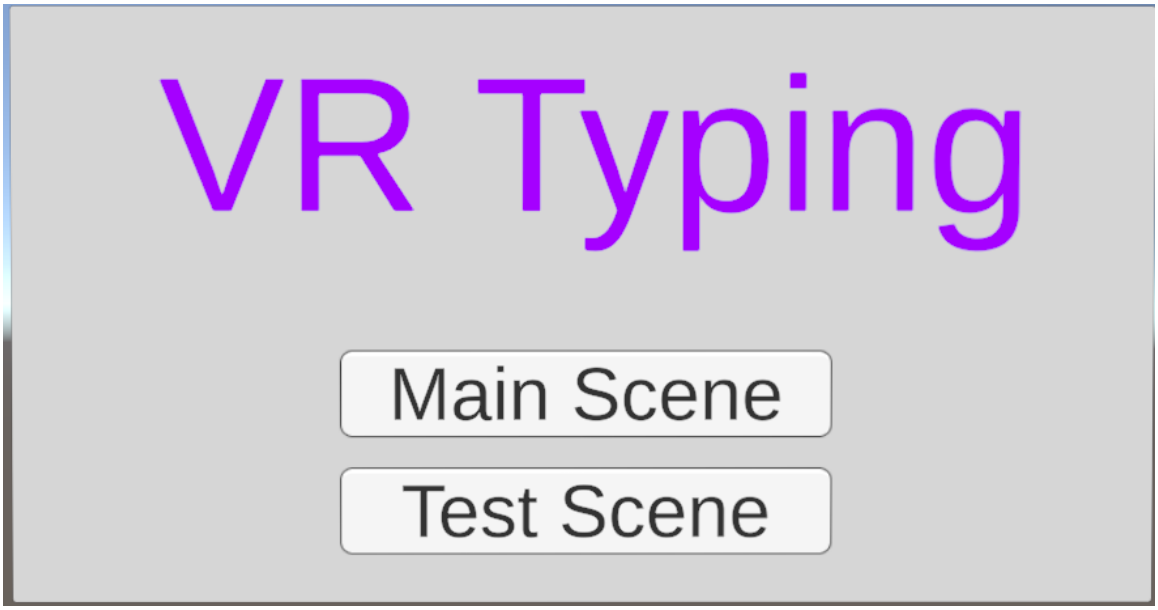


Figure 6.2: The menu scene

The second scene is the "Test Scene" which allows the user to use both the dictation and the drum-like keyboard input methods freely. This scene also displays to the user what their current words per minute (WPM), characters per minute (CPM), and errors per minute (EPM) are. Lastly, this scene displays a rotating list of sample texts that are not included in the actual experiment. This scene is to allow the user to test and play around with the input methods to get used to them before the experiment. This lowers the amount of learning that has to happen inside of the actual experiment and will allow those who aren't familiar with VR to get accustomed to it as well as the input methods. To allow the user to start the experiment, there is also a button that will take the user back to the "Menu Scene". The mechanism for pressing this button is the same as in the "drum-like keyboard". This scene can be viewed in Figure 6.3.

All WPM in this work is calculated using the following formula [62]:

$$WPM = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5}$$

where, $|T|$ is the overall resulting length of an input string, S is the number of

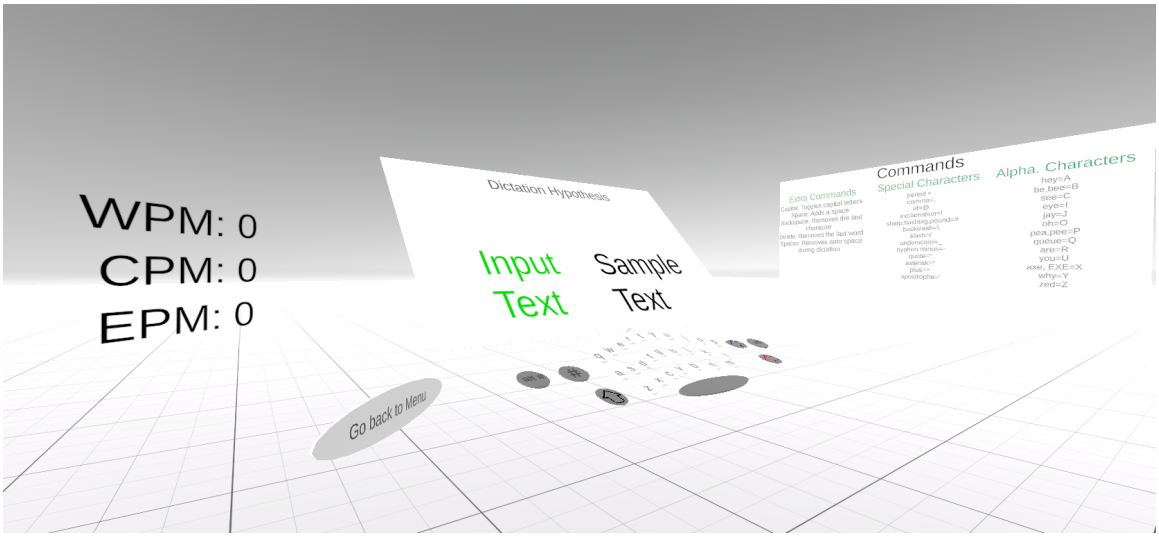


Figure 6.3: The testing scene

seconds between the first and last input. $|T|$ is subtracted by 1 due to the fact that time does not start until the first input. The formula multiplies by 60 to transfer from words per second to words per minute, and it divides by 5 to transfer from characters per minute to words per minute, since a general word is considered to be 5 characters.

The third scene is the "Main Scene" this is the scene where the actual experiment takes place. This scene is very similar to the "Test Scene", except it has three main differences. The first difference is that the WPM, CPM, and EPM readings are not shown to the user and are instead averaged together per stage. Each time the user finishes a stage with an input method, this average gets output to a file and reset for the next stage/input method. This was done as a way of getting the user to try their best without any thoughts or anxiety about not performing as well as they did during the test scene. The second difference is that the return to menu button doesn't exist, so that the user can't accidentally hit it. The final difference is that the input methods rotate and the sample text is taken from a wider list of possible texts that do not include any sample text from the test scene. This scene can be viewed in Figure 6.4.

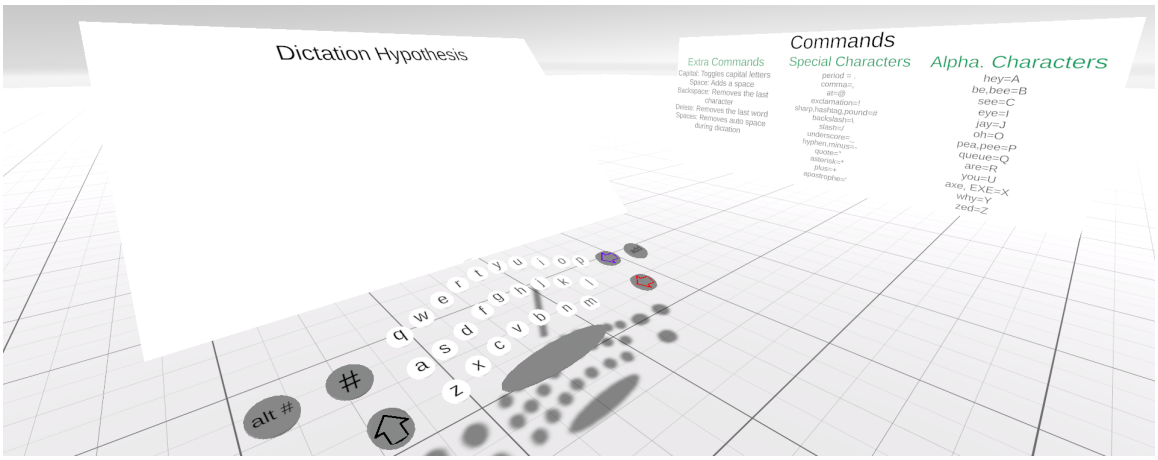


Figure 6.4: The main scene

6.3.6 Typing Text

The text that the participants had to copy involved four types of input text. This text was made out of four categories: URLs, email addresses, sentences, and paragraphs. The reasoning behind using URLs is the fact that they are very common when using a 2D typing interface, and most people are familiar with them. They also lend an interesting problem set to this process. URLs tend to need special characters, non-words, and specific abbreviations to function. This includes abbreviations like www, com, net, org, and edu. Special characters exist in URLs as well, ".", "/", and, ":". Emails also contain special characters, non-words, and abbreviations. They are also very common in the use of 2D typing interfaces. Sentences and paragraphs can contain a wider set of special characters than emails and URLs and can also test the effectiveness of typing methods for longer periods than emails or URLs. This means that both the verbal and physical input methods have to account for most/all special characters, non-words, and abbreviations. With these four input types, we believe that every use case for typing in VR and in 2D interfaces can be tested.

The sample text that is being pulled into the experiment are generated using online tools to verify consistent, or almost consistent, lengths and complexity. This text is placed into separate files to be pulled in at run-time when the user reaches any specific stage or completes the stage with one of the input methods. The input

type and it's associated average character length and standard deviation is found in Table 6.3.

Table 6.3: Each input type with their associated average character length and standard deviation

Input Type	**Character Length Mean (SD)**
URL	10.2 (1.7)
Email	17.2 (2.2)
Sentence	37.5 (7.8)
Paragraph	305 (14.2)

6.4 Results

As seen in Table 6.4, both input methods implemented in this work have quite different words per minute (WPM) averages for each type of input. Paragraphs and sentences for both input methods have larger WPM averages and larger standard deviations, drastically larger in the case of the dictation + drum-like keyboard input method. The email and URL input types had very similar results for both input methods.

Table 6.4: The mean WPM, with standard deviation, and WPM Range for each of the input methods in each input type.

Input Method	**Input Type**	**WPM Mean (SD)**	**WPM Range**
Dictation	URL	7.83 (4.31)	2.47 - 17.26
	Email	5.00(3.79)	0.99 - 14.08
	Sentence	12.17(6.02)	2.414 - 19.5
	Paragraph	28.08(20.15)	10.38 - 69.11
Dictation + Drum-like keyboard	URL	7.70(4.71)	5.03 - 18.42
	Email	5.40(3.49)	2.34 - 14.45
	Sentence	18.83(13.35)	3.556 - 41.8
	Paragraph	31.69(19.28)	13.66 - 66.79

As seen in Table 6.5, the mean errors per minute (EPM) found for each input type are very similar to each other within each individual input method. The dictation + drum-like keyboard input method does have an average error rate and standard

Table 6.5: The mean EPM, with standard deviation, and EPM Range for each of the input methods in each input type.

Input Method	**Input Type**	**EPM Mean (SD)**	**EPM Range**
Dictation	URL	2.09 (2.02)	0.21 - 6.6
	Email	2.22(1.51)	0.6 - 5.14
	Sentence	2.41(1.78)	0.21 - 5.97
	Paragraph	3.22(2.51)	0.76 - 8.75
Dictation + Drum-like keyboard	URL	4.00(4.05)	0.22 - 13.45
	Email	3.40(3.15)	0.33 - 10.00
	Sentence	4.32(4.14)	0.55 - 10.06
	Paragraph	5.72(3.87)	1.86 - 16.38

deviation of about twice that found with just dictation. The maximum errors per minute for this input method are also about double that of the dictation input method. The minimums are a bit different. The minimum errors per minute for the sentence and paragraph input types share the doubling found with the maximum, mean, and standard deviation; however, the email input type for the dictation + drum-like keyboard input method is about half that found in the dictation input method. The URL minimums are about the same.

The System Usability Scale (SUS) questionnaire is a measure of the usability of each system. The mean SUS score can be found in Table 6.6. These scores range from 0 to 100. The SUS was given to each participant twice. Once for the dictation input method and once for the dictation + drum-like keyboard input method. This table showcases that, on average, users found the combination input method of dictation + drum-like keyboard more usable. To give a better grasp on the data collected, Figure 6.5 and Figure 6.6 are included. Figure 6.5 showcases the SUS scores for the dictation input method. Figure 6.6 showcases the SUS scores for the dictation and

Table 6.6: The mean SUS scores and SUS range

Input Methods	**SUS Mean (SD)**	**SUS Range**
Dictation	54.63 (26.54)	12.5-100
Dictation + Drum-like Keyboard	68.75 (16.81)	37.5-95

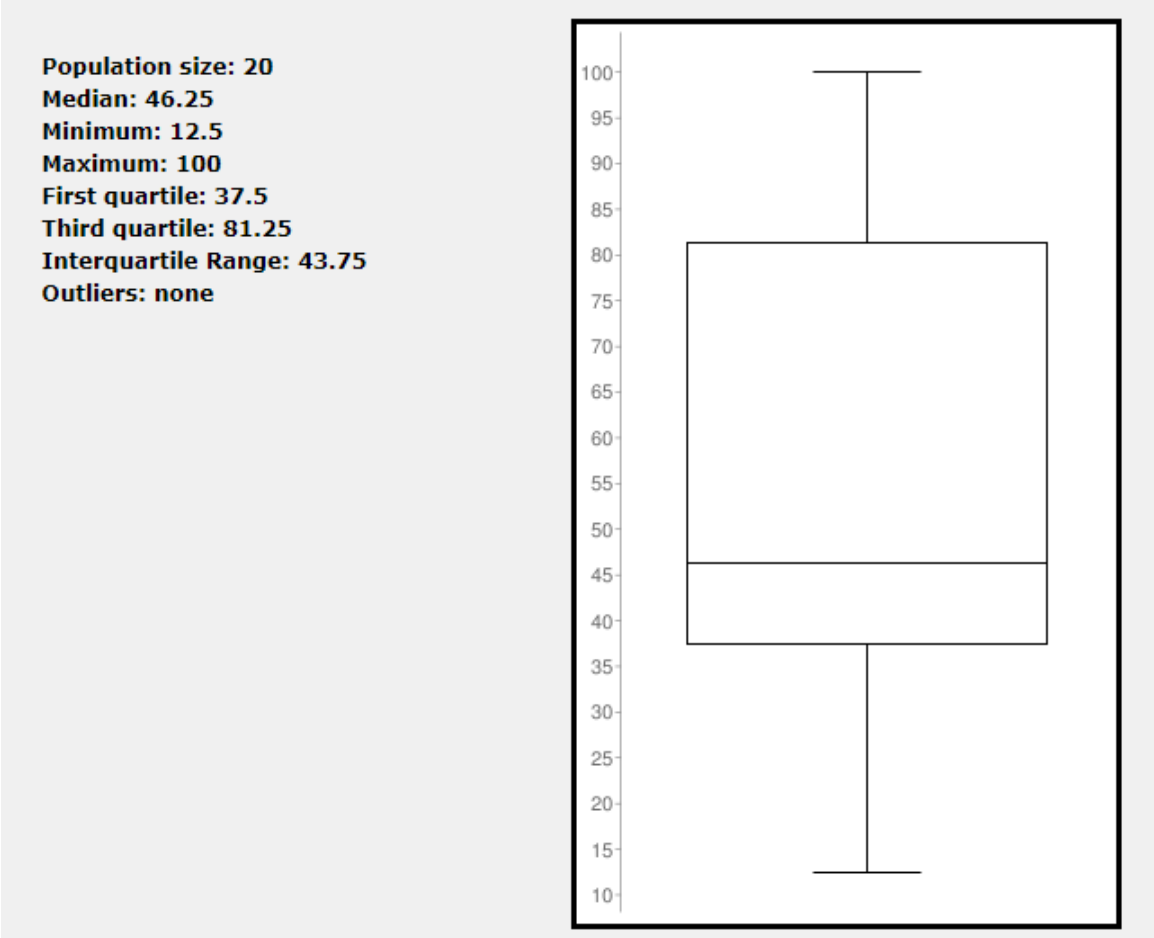


Figure 6.5: The box and whisker plot of SUS scores for the dictation input method

drum-like keyboard input method.

The participants were also given a post-survey. This post-survey consisted of ten questions, which can be seen in Figure 6.7. The first five questions were on a Likert scale from one to five. Participant responses to these five questions can be found in Figure 6.8. Users reported mostly five out of five for question one, comfortability during the test. Users reported mostly four out of five for question two, ease of text input. Users reported mostly four out of five for question three, intuitiveness of the user interface. Users reported mostly four out of five for question four, usefulness of the typing methods. Finally, users reported mostly five out of five for question five, overall experience.

Questions six to ten were free response and were then analyzed for significant

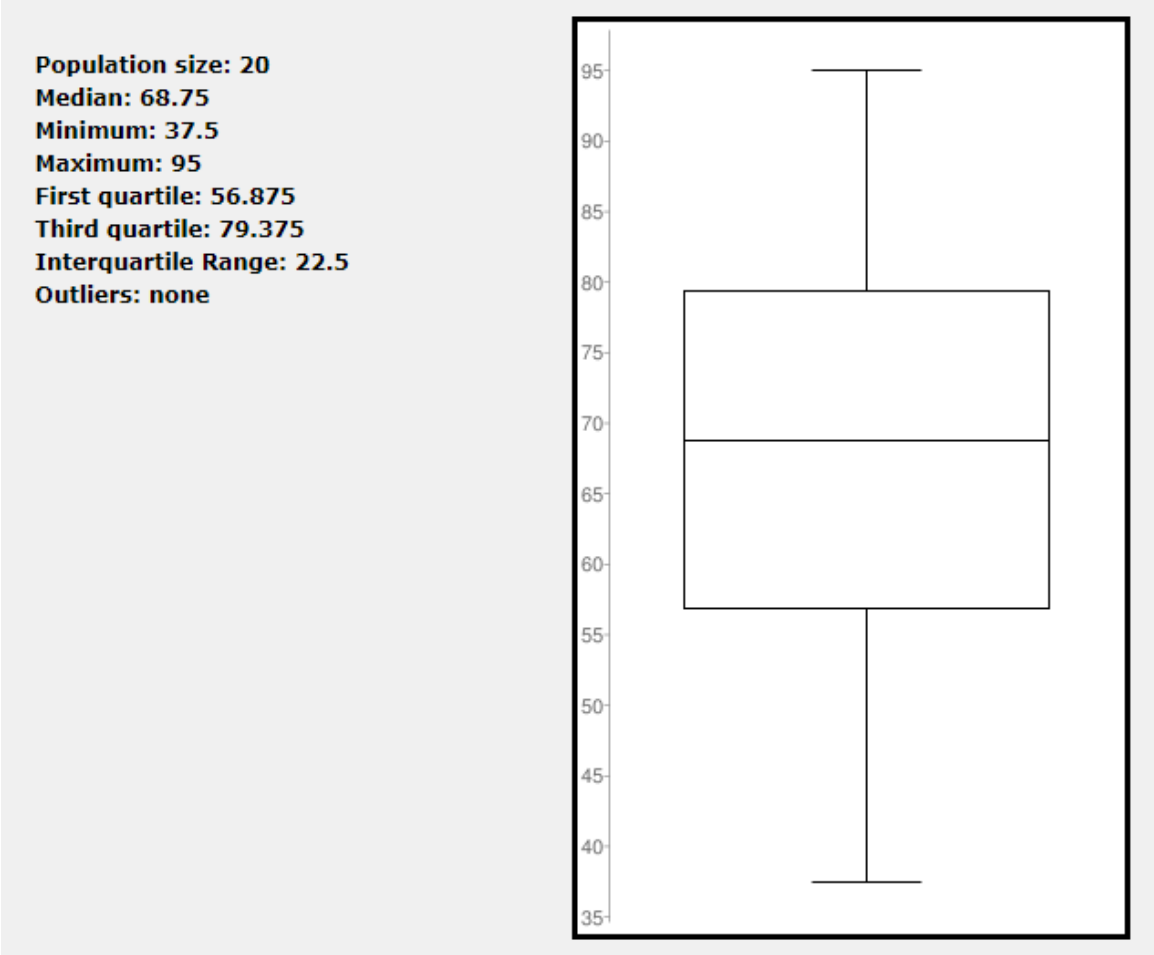


Figure 6.6: The box and whisker plot of SUS scores for the dictation + drum-like keyboard input method

Q1	Please rate how comfortable you were during the test
Q2	Please rate how easy it was to input each text
Q3	Please rate the intuitiveness of the user interface
Q4	Please rate the usefulness of the typing methods in general
Q5	Please rate your overall experience
Q6	What potential improvements, if any, would make the application more useful or easy to use?
Q7	Based on your experience, what are some other virtual reality typing methods you would find useful?
Q8	Based on your experience, do you have a preference for any typing method?
Q9	Based on your experience, do you think any typing method was better for one input type or another (email, url, sentence, or paragraph)?
Q10	Please write any other comments or observations that you might have.

Figure 6.7: The questions asked during the post-survey

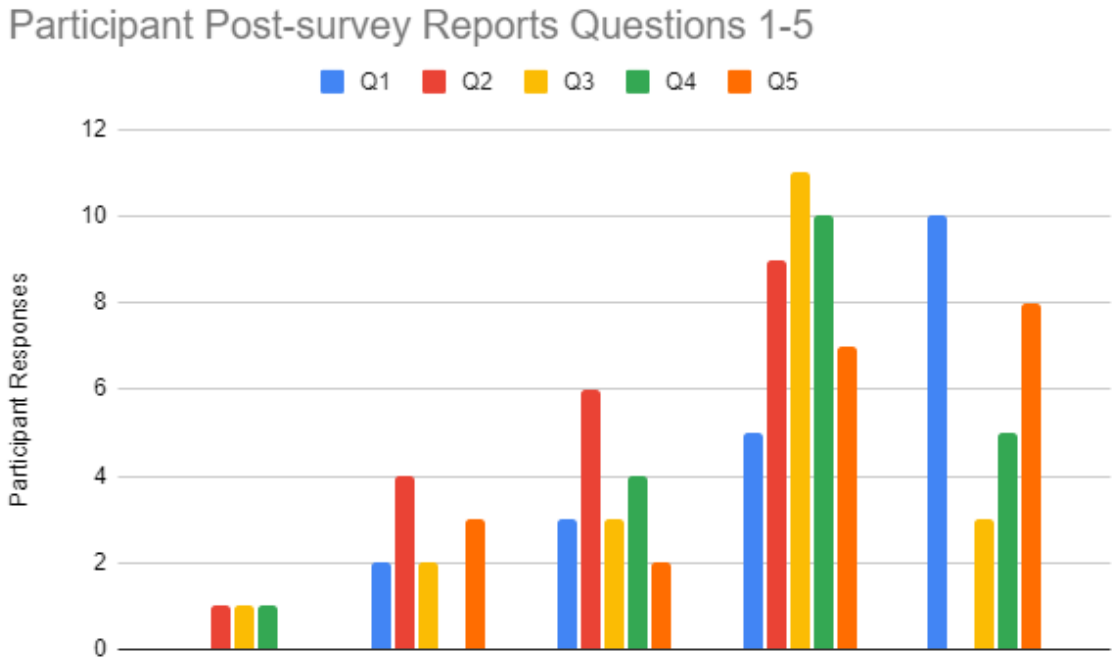


Figure 6.8: Participant responses for questions one through five of the post-survey

statements and positive/negative sentiments for each input method and for each input method on each input type. Dictation had a total of fifteen positive statements made about it and nine negative statements. There were also eleven statements about the need to clarify or indicate some aspects of the system. The dictation method was also described as good for sentences and paragraphs, with thirteen and seventeen statements respectively. URLs and Emails received three and four positive statements respectively, for the dictation method. The drum keyboard had a total of thirteen positive statements made about it and six negative statements. There were also five statements asking for clarification or improvements in the system. The drum keyboard was described as useful for URLs and Emails, with eleven and twelve responses respectively. The drum keyboard also received two positive responses for sentences and three positive responses for paragraphs. Many of the statements in questions six to ten were responses that described the outcomes above, and many other statements gave feedback as to what needed to change and what other input methods they'd like to see.

Lastly, participants were given the Simulator Sickness Questionnaire (SSQ) to determine if typing in VR using these methods gave any excess simulator sickness. It was determined that this application does not give any excess simulator sickness, with the most common and severe symptom being eye strain and averaging to be less than mild eye strain, which is an average of less than one out of four.

6.5 Discussion

6.5.1 Demographics

Demographic data was taken directly from the pre-survey given to the participants before they entered into the VR typing application. The total number of participants was twenty. The median, and mean, age of the participants was thirty-one years of age. A total of fifteen males and five females took part in the study. The majority of participants have completed a bachelors degree. On a scale of one to five, the majority of participants were somewhat familiar with VR and responded with a three out of five or a four out of five. The exact values for this question can be seen in Figure 6.9. The participants were also asked, on a one to five scale, if they were familiar with motion tracking. The responses were more scattered, but most people rated a one out of five. This response shows that most participants didn't have much familiarity with typing in VR, as most VR typing methods rely on motion tracking. Lastly, the participants were asked if they were familiar with electronic entertainment and most participants rated a four, or higher, out of five, with the large majority answering five out of five.

6.5.2 Performance of Each Method

Each method performed extremely differently when comparing across input types. Many participants reported that dictation really helped with longer sentences and paragraphs, as long as they were pretty simple. The largest factor for speed in sentences and paragraphs, as well as error rate, was whether or not the dictation al-

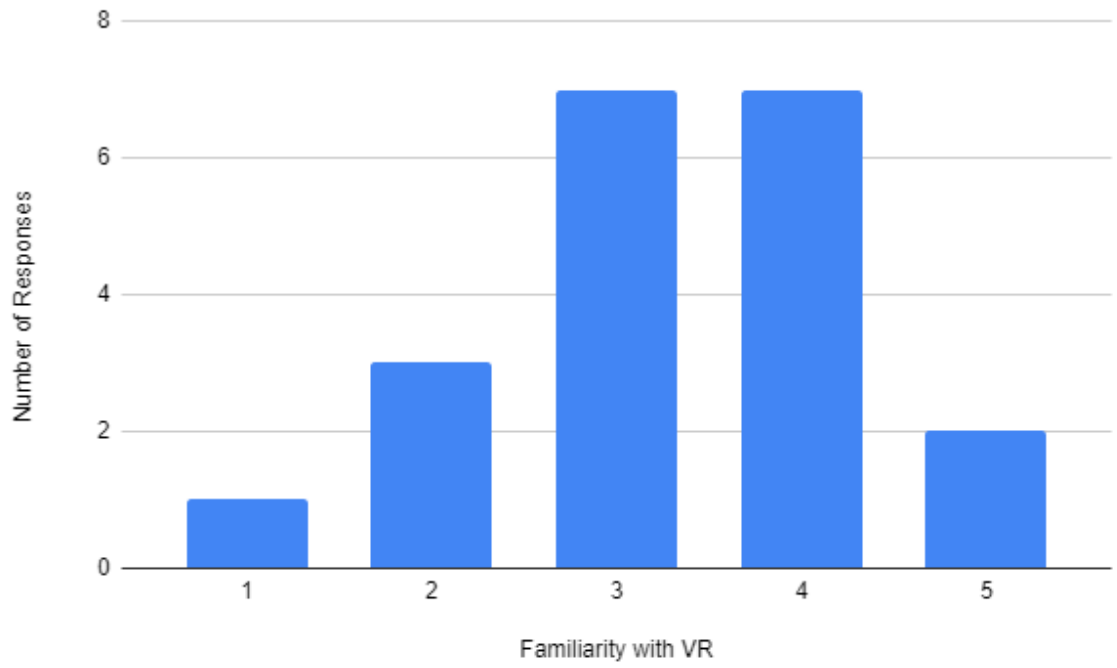


Figure 6.9: Participant responses for familiarity with VR

gorithm understood what the participant was saying. Many participants experienced unintended results due to the dictation results not being accurate. A large factor that hindered speed in all input types is the need to specify commands to change how the system worked. This includes commands like: "Command Spaces" and "Command Capital". A few participants reported during their post-survey that using the commands felt cumbersome, and that inputting commands fast caused the dictation result to be less accurate due to them no longer overpronouncing their dictation. Many participants also responded with the need for auto-capitalization in sentences and paragraphs. While this doesn't solve all of the problems that dictation has, it would improve accuracy, speed, and usability to a significant degree.

Participants often responded very well to the dictation and keyboard combined method. Interestingly, despite the increased usability scores and the specific mentions of enjoyment found in the post-survey, this combined input method resulted in worse errors per minute, but still faster average words per minute than the dictation method.

Another comment found in the post-survey was in the need to have more time

with the systems. In particular, users found that they didn't have a large grasp of what words the dictation algorithm would insert easily and which words would require large amounts of overpronouncing. This is a common issue found with dictation algorithms as all of them, and conversational agents as a whole, can't directly tell you what they're good at. It requires a great deal of practice and trial and error to fully understand the use cases and abilities for each dictation algorithm.

Each participant was put into the test scene and were told to arbitrarily leave it when they found that they had practiced enough and had a good grasp of the input methods. This resulted in many users spending more time in the test scene than the main scene. Despite this practice; however, there was found to be no statistical correlation between time spent in the test scene and performance. For completeness, Figure 6.10 shows the time spent in both the test and main scenes per participant, as well as average and standard deviation for time spent in both scenes.

6.6 Conclusions

The dictation and drum-like keyboard input method was, overall, faster and more prone to errors than the dictation input method. Many users found the drum-like keyboard fun to use in short bursts, but over a larger use period the method was found to be cumbersome. This was due to the large amounts of movement required in typing anything lengthy.

Differing input types were found to change the average speed of both methods considerably. Dictation was determined by users to be better for longer strings of real words, like sentences and paragraphs. The drum-like keyboard was determined by users to be better for precise phrases or special characters.

Due to the large discrepancy found in speeds and error rates for each input type, it is our recommendation that future work regarding text input in VR should include not only sentences or phrases, as commonly found in current research, but also: paragraphs to test long form and endurance writing, and email addresses or URLs to test short input and special character insertion. More input types can be added to fully

P#	Time Spent in Test Scene	Time in Main Scene
1	3.40	6.00
2	7.03	9.31
3	23.26	16.32
4	5.03	7.58
5	14.08	5.37
6	7.02	4.43
7	8.24	14.56
8	6.03	12.38
9	12.06	6.09
10	14.57	8.02
11	8.51	10.25
12	16.57	8.02
13	4.02	4.22
14	18.48	30.15
15	4.11	25.14
16	10.59	6.17
17	18.57	12.42
18	5.58	6.34
19	7.34	14.35
20	9.25	3.35
Total Time	203.74	210.47
Avg Time	10.48	9.03
Standard Deviation	5.76	3.73

Figure 6.10: The time spent in each scene per participant

encompass everything a user might type with a traditional keyboard. Overall, the addition of more varieties of text in researching typing methods is incredibly important as VR transitions from a novelty entertainment and scientific games platform, to an interface that many people can use daily in their work.

6.7 Future Work

Typing in VR has always been a rather slow and non-portable process. As part of the research to fix these issues, we would like to create a generalized interface system. This system would allow the user to move around the keyboard, or any other object associated with it, in 3D space complete with six degrees of freedom. This system would also allow the user to scale and disable the interface at will. This system, therefore, would be extremely useful in applications and other research that involves both locomotion and typing intermittently.

We would also like to incorporate many input methods into this research to allow a more complete observation of input methods, particularly in the case of the SUS. With a full view of the available input methods, we believe that participants of the user study would rate each method differently on the SUS.

A longitudinal study incorporating all of the input methods would indicate what the average speed of each input method actually is. Users in this study, for instance, displayed a need to use the input methods more to fully understand the methods. Some users reported that they didn't fully understand the input methods until some of the longer input types, or towards the very end of the study. Typing in 3D is not something that most users have experienced, so letting them get used to the systems is imperative.

Compound commands are when a participant would aim to give two commands at the same time to the dictation system. This was a somewhat common mistake users would feel natural in doing. These compound commands often were caused by the want/need to have uppercase or lowercase letters. The compound commands would take the form of ""CommandWord" Capital letter", where letter is a letter

they wanted to input. The proper syntax in the current system to accomplish the same thing would be, ""CommandWord" Capital" + ""CommandWord" letter". The new type of capitalization system is almost analogous to a shift key opposed to the current system which is analogous to a caps lock toggle. Implementing the compound command system would severely increase spelling speed and severely lower error rates in dictation.

With the addition of input methods, we would like to incorporate all of them into a typing game/test. This VR typing game would be complete with a virtual environment, background music, locomotion, leaderboards for each input method, multiplayer competitive scenarios, and non-player character battles.

We also feel that there can be more creative ways to allow the user to type and change input types. A few examples of extra features to incorporate into the typing for the drum-like keyboard are: using the radial touch pad on the VIVE controllers to change between keysets; colliding both controllers on a single key to change the input without completely changing the keyset; allowing the user to change, create, and save keysets dynamically; and allowing the user to use the SWYPE feature, commonly found on mobile devices, using the drum keyboard as to minimize the amount of movement per word and increase user endurance. There are also creative ways to add commands to dictation that might shorten the amount of typing necessary. A good example of this is adapting the dictation algorithm to automatically structure if statements or other programming structures, thus allowing for less key strokes and faster implementation.

6.8 Acknowledgements

This material is based in part upon work supported by the National Science Foundation under grant numbers 019609 and IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Chapter 7

Game Mechanics

7.1 Introduction

With the addition of typing in VR to this work, we felt that expanding into a more game related field would be appropriate. A pretty large demographic of people have played games related to typing in either competitive settings or simply against themselves. Whether racing horses, cars, motorcycles, or animals the games are designed to try and push the user into typing as fast as they can with the smallest amount of errors possible. We applied this same thought process in the creation of this game, with one key difference. A virtual environment that the user can explore while waiting for connections to games, or just generally enjoy while they play. This chapter also allows us to discuss some interesting practical topics about Unity or specific aspects of game design and natural phenomenon.

The layout of this chapter is as follows: Section 7.2 is dedicated to game attributes including, multiplayer, input methods, locomotion, and non-player character(NPC) design; Section 7.3 is dedicated to ambiance needed for virtual environments to function and flourish; and Section 7.4 is dedicated to future expansions to this applications that, given more time, would further this application and general VR applications.

7.2 Game Attributes

7.2.1 Leaderboards

Leaderboards have been a staple in video games, but the most prevalent use of leaderboards has been in arcade-based games. These leaderboards indicate to any user how well they did compared to their local community and can allow a competitive zeal towards a game. Competing for high scores is an incredibly common trend in all arcades, where people would vie for top spots and spend hours, and money, playing the arcade games. This trend has continued in the current video game space, albeit in a different way. Video game players no longer wish to spend money on each attempt at a game, so repeat attempts are generally free after buying the game as a whole. Leaderboards in this type of environment can be based on a friends list, regional list, or global list. This means that scores are stored in accessible servers around the world. The major downside of this approach; however, is caused by the cost that companies need to pay to keep these servers up for long periods of time. Video games can now, generally, be played at any place at any time for years and years after their release date. This poses a difficult question for game companies. When do you turn the servers off? Doing this at all saddens fans and, in some cases, makes games completely unplayable. When this happens, leaderboards are then worthless as they usually only track one of each user's score to send to the leaderboard until it gets replaced with a better score. This is why a localized approach is necessary to build in to games that want to keep score. Leaderboards should keep multiple scores and allow the user to associate any of them with dynamic names. This approach can always be on, where the game still only sends the highest score to the servers, but when the servers go down there needs to be a back-up system already in place that still has all of the user's scores from before the servers went down. These leaderboards could also include the last known highscores of the users in the current user's friend list from before the servers went down, to give benchmarks and a sense of nostalgia.

This game uses a simple localized leaderboard, one for each input method. An

example of this leaderboard can be seen in Figure 7.1 Each input method needs to have it's own leaderboard since some are inherently faster than others. Score is calculated by: $WPM - EPM^2 = Score$. The reason for this is to make each error worth more than the last, but still give some leeway to make errors. There is an uncountable infinite number of ways someone could calculate score given a float coefficient.

Figure 7.1: A sample leaderboard that appears when the race is over

7.2.2 Input Methods

Different typing input methods have already been identified and discussed in detail in Section 6.2. This application implements the drum-like keyboard, dictation, point and select, and two-thumb touchpad input methods and each can be seen implemented in Figure 7.2, Figure 7.3, Figure 7.4, and Figure 7.5 respectively. These four input

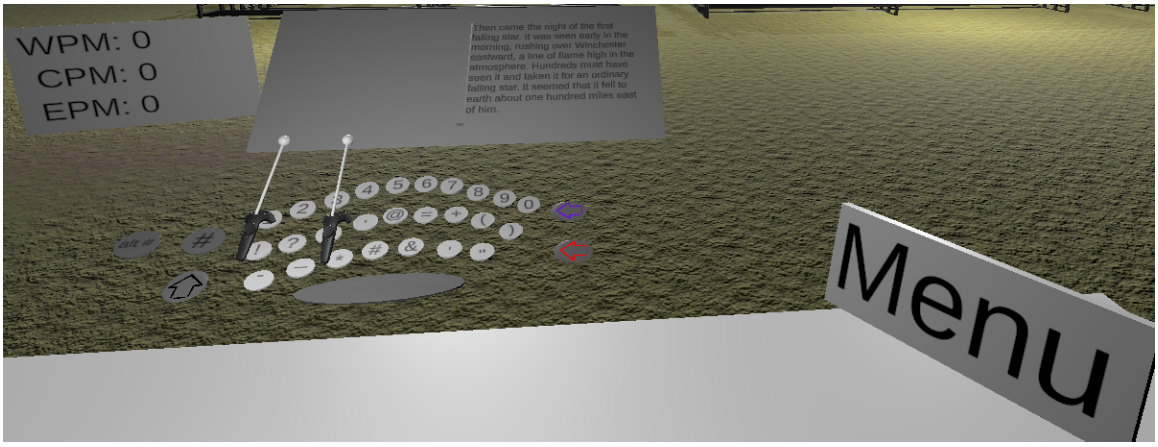


Figure 7.2: The drum-like keyboard typing method implemented in-game

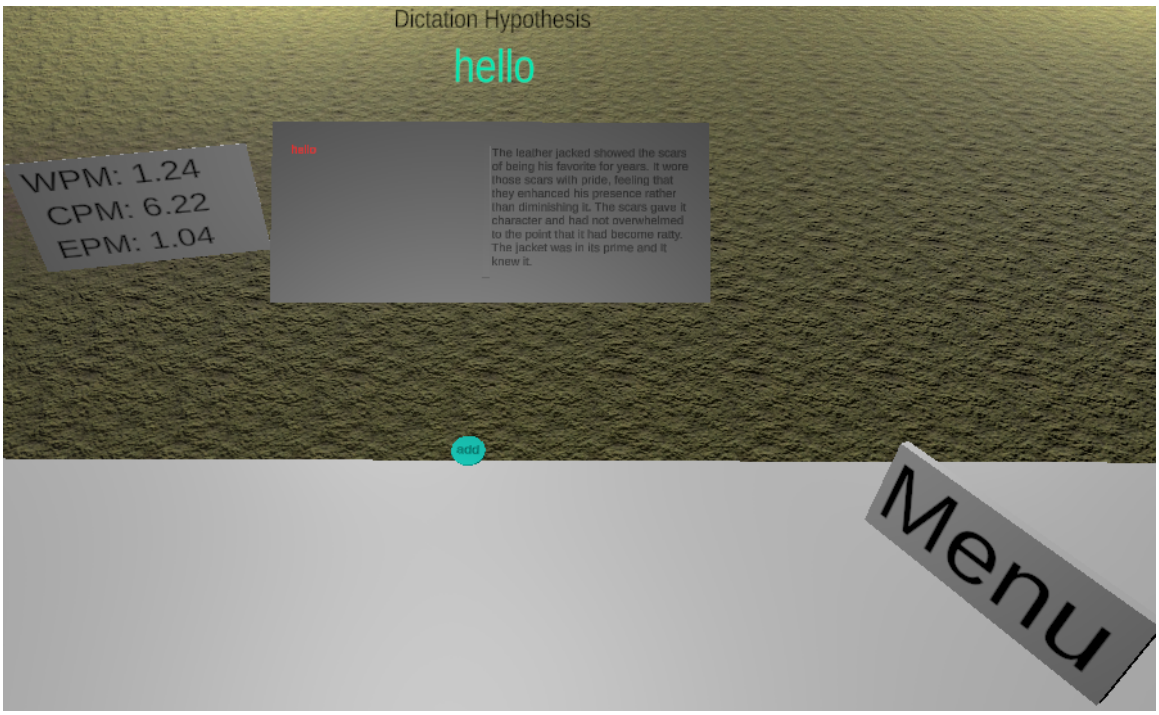


Figure 7.3: The dictation typing method implemented in-game

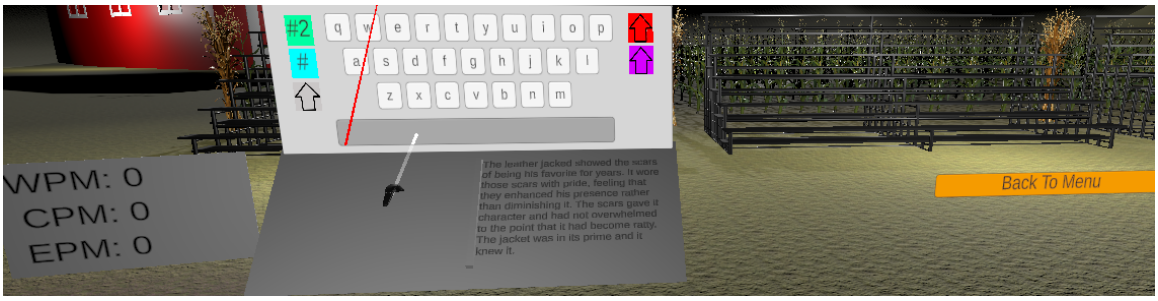


Figure 7.4: The Point and Select typing method implemented in-game

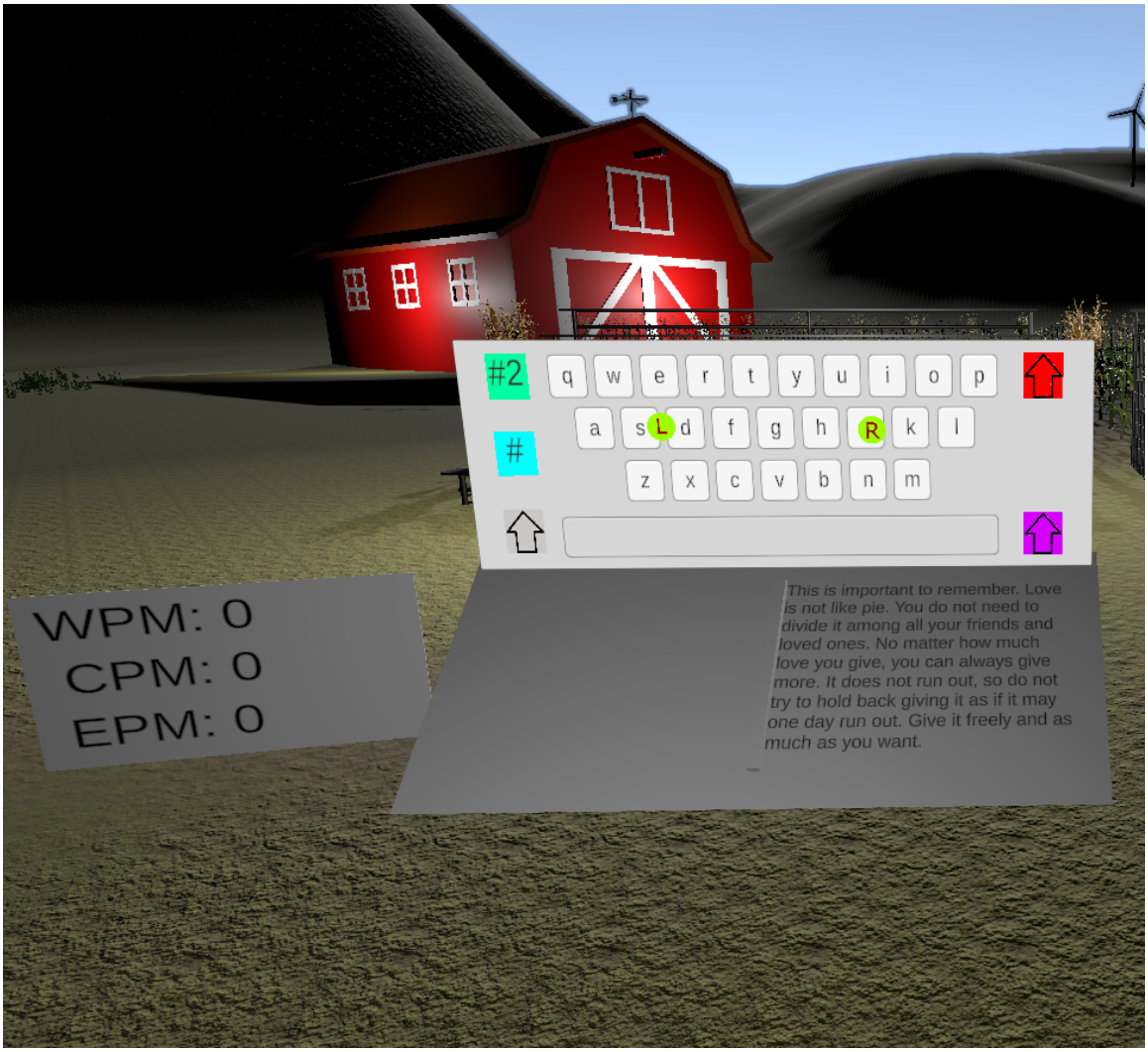


Figure 7.5: The Two-Thumb touchpad / Split keyboard typing method implemented in-game

methods make up a large portion of the identified input methods. They were implemented due to the fact that they were determined to be the fastest four input methods that have been identified in the literature. Punch-typing is very similar to drum-like typing, but it's a slower input method. Dwell and gaze typing are also very slow, though they do provide a level of accessibility that should still be heavily considered. All of these input methods are future work, so that the project can display all of the identified input methods in literature.

7.2.3 Locomotion

Different locomotion techniques have already been identified and discussed in detail in Section 2.1. This application uses Joystick movement, where the user simply slides in an indicated direction, at a specific magnitude. To reiterate, this input method does not require a joystick or trackpad to operate correctly. This locomotion method simply requires an actuator and a vector. The vector's magnitude will always be set programmatically to a constant in this application. The actuator for this movement is different based on the different input methods. If the input method does not use the trackpad at all, the locomotion will be operated using the trackpad as both actuator and angle of the vector. If the input method does require the trackpad, like in the case of the split keyboard [94], then the trigger is the actuator and the angle is set based off of the rotation of the controller. Only yaw rotation (y-axis rotation in Unity) of the controller is taken into account for the angle of the vector. The user should only move in the XZ plane and have no verticality. Both of these techniques could use the distance between the controller and the user's torso (or XZ coordinates of both in Unity) to determine speed/magnitude, but it was determined to not be necessary, despite giving the user more control over their movement. This is future work.

7.2.4 Graphical User Interfaces

Developing for VR has a few extra items to consider compared to other virtual environments. The main difference is in how a developer needs to manage graphical user

interfaces (GUIs). User interfaces in traditional 2D and 3D applications can largely be always on or displayed most of the time. In VR, the user can't usually access GUI elements that are 2D, so all of them are generally in 3D. Developers could make use of the trackpad or joystick on controllers to select and move between 2D interfaces, but that isn't very common. It's also counter intuitive to have GUI elements covering the user's field of view in these 3D virtual environments, as one of the reasons to use VR is for the sense of presence it brings. This type of GUI is also unrealistic, further taking the user out of the environment and ruining their sense of immersion. Not every application has to be immersive or realistic, however.

One school of thought for realism in VR is in having no GUI. This means that any modifications or anything else that the GUI would normally interact with gets put into a 3D button or other interactable object that is actually inside of the virtual environment. This approach isn't the most practical as the user won't always have access to this interactable. This could be solved by making an inventory system of some kind without GUI elements. This is done by allowing the user to store and switch between objects by gesturing in a certain way. For example, while the game "Space Pirate Trainer" [47] has a GUI, it also has an inventory system for the weapons where the user puts the controller behind their back and the weapon will automatically switch. If a developer does this with buttons or other objects to do the same thing, this makes a no GUI system possible, but not very practical.

The next school of thought is in using 3D GUI elements that are always on, but have customizable transparency. These GUI elements often come with some sort of settings menu and are placed in areas of the application that aren't often used. This approach can be seen in Figure 7.6. This approach is not at all immersive and is only useful in specific applications, where the user doesn't generally have to locomote.

The next GUI technique is full gesture-based display. This GUI technique displays specific GUI elements based off of gestures. For example, Figure 7.7 displays gesture based GUI when controllers are pointed specific ways. This type of GUI is more immersive than all of the other methods, besides the no GUI method, but it

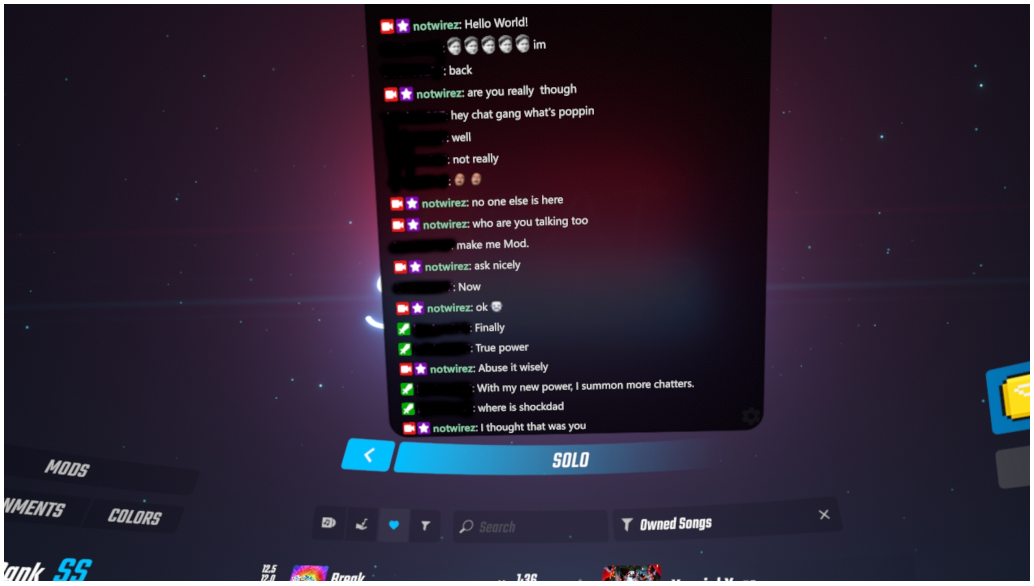


Figure 7.6: GUI integration of Amazon's streaming service "Twitch" [4] in the VR game "Beat Saber" [28]. Image captured from a live Twitch channel [3]

also allows a useful GUI to be displayed. This technique strikes a balance between practicality and immersion. A very similar GUI technique is button-based display. In this technique the display mechanism is activated by pushing a button on the controller. This is currently more useful than the gesture-based display technique due to the fact that gestures are hard to recognize. Due to this, most applications that use gesture-based displays only allow one or two easy to replicate gestures. A button based approach can be quite varied, especially if taking into account the trackpad or joystick feature as this can be used for a pseudo-infinite selection based GUI. This selection based GUI is essentially a spiral that only shows the current layer, but will move the layer down as the user progresses through it. This GUI can be seen in Figure 7.8, even though this is implemented in a desktop application, it could be adapted to work in VR with the trackpad.

There are a few other concerns when making GUI in VR, particularly when dealing with text. Text close up can be of almost any color, but the farther the user's distance is to the text, the more the text seems to blur together with specific colors. Both really light (neon) colors and really dark colors can cause these issues,

Figure 7.7: GUI integration of Amazon's streaming service "Twitch" [4] in Valve's VR game "Half-life: Alyx" [105]. (A): Twitch GUI is not displayed when user rotates controller to face the user. (B): Twitch GUI is displayed when user rotates controller away from the user. Image captured from a live Twitch channel [3]

Figure 7.8: Spiral GUI from Digital Extreme's "Warframe" [22] (A): The beginning of the pseudo-infinite spiral (B): Another section of the spiral that still shows some items from the beginning of the spiral, for reference. Image captured from the author's account.

despite providing high contrast. Dark colors are usually okay at a distance, however. The main issue with this blurring is caused by a lack of kerning, or horizontal space between letters. This issue can cause issues like the string "rn" looking like the letter "m", particularly at a distance. Another issue is caused by a lack of backdrop behind the text. This can cause text to be unreadable against the general virtual environment. Generally, if there is text in VR, to get the text as readable as possible, use dark colored text on light colored, but not transparent, backgrounds. Try to keep the text to sans serif fonts too, as these fonts are more friendly to users with dyslexia, and make text less crowded, so it's easier to see at a distance.

If a developer chooses to use a 2D GUI in VR with event system based controls, it is important to heavily test the GUI in VR. Development tools, like Unity, show more in their development views than the VR headset may be able to. Thus, the developer's GUI can go outside of the user's FOV very easily, cutting off the GUI at the corners and maybe even chopping off words. This 2D GUI should also not cover the entire screen as any GUI like this will, by default, stick to the user's FOV. This can cause VR sickness if it covers the whole screen due to the fact that it looks to the user like nothing is changing when they rotate their head. The whole screen GUI also makes text and objects incredibly hard to see at the edge of the user's vision. Generally, 3D interfaces are rotated to face the user to solve this issue.

This application uses simple 3D GUIs to indicate to the user when inputs are done correctly, what their WPM and EPM is like, if they're winning a race, or how they compare on their own leaderboard. These GUIs aren't realistic, but they are useful and they are commonly found in applications similar to this type of application. Examples of the menu GUI and leaderboard GUI can be seen in Figure 7.9 and Figure 7.1. There is also a GUI associated with returning from the play scene back into the menu scene. This GUI comes in two forms: a simple 3D GUI and a 3D object. The 3D GUI version is used in input methods that interact with the GUI Keyboard, the "Point and Select" and the "Two-Thumb Touchpad" input methods. The 3D object is used when the drum controllers are being used, the "Drum-like

Keyboard" and the "Dictation" input methods. The difference is due to the fact that the 3D object allows the drumstick to collide with itself, which can then trigger it's function, while the 3D GUI can implement collisions, the GUI would be too far away for the user to reach over in cases where the drum controller is being used. These GUIs can be seen in Figure 7.10.

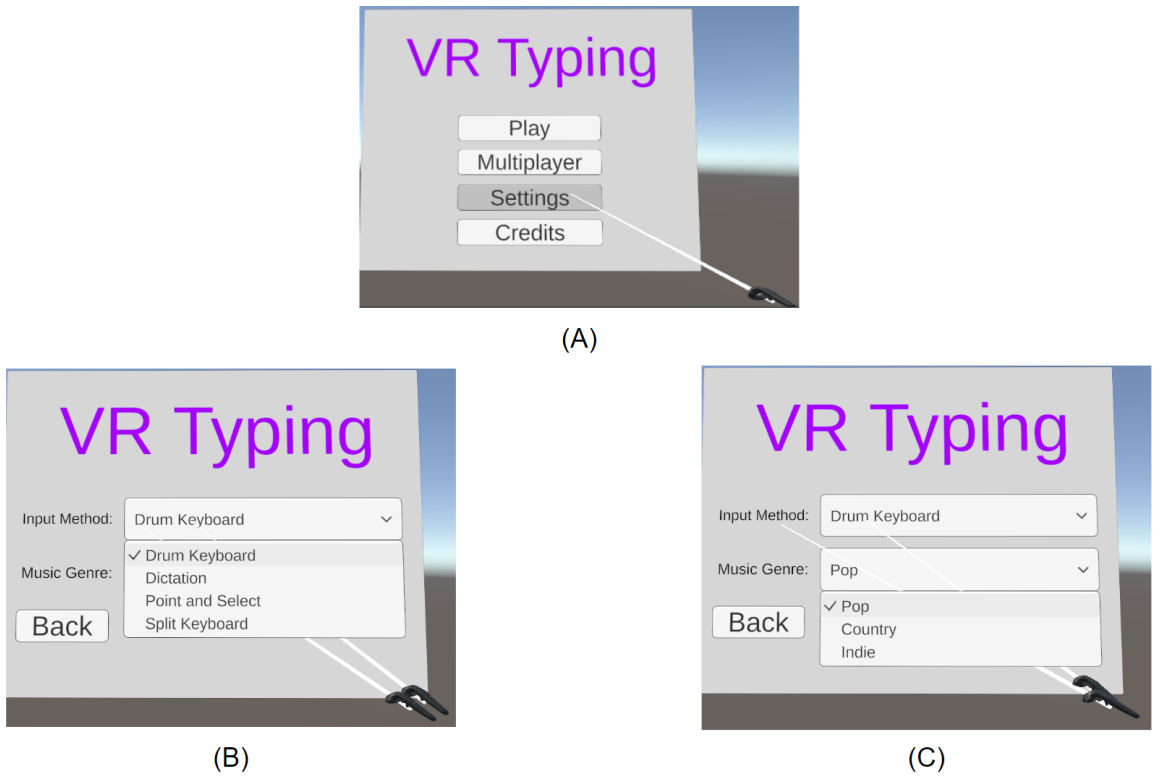


Figure 7.9: The initial menu for the application

Figure 7.10: (A): The 3D object GUI to send the user back to the menu. (B): The 3D GUI version

7.2.5 Non-Player Characters

Non-Player Characters (NPC) are integral to many, if not most, games. Therefore, it is very important for this work to cover their use and how to implement them in VR.

7.2.5.1 Models

Models for NPCs in VR are, generally, required to be 3D. A developer could get away with using 2D models if the user is not allowed to locomote too heavily in VR, but this project encourages the use of locomotion, and thus all of the models for the NPCs are in 3D. Modeling is generally not encouraged in Unity as, by default, Unity only allows primitive 3D objects to be incorporated into models. Unity primitives are: cube, spheres, cylinders, capsules, planes, and quads. Amazing things can be done with these primitives, but using 3D modeling software, such as Blender [26], developers and artists can create truly gorgeous and intricate models. Unity also allows importing 3D models of the following file types: `.fbx`, `.obj`, `.dae`, `.3ds`, and `.dxf`. Unity also allows the following formats if the corresponding modeling software is installed on the developer's PC: `.ma`, `.mb`, `.max`, `.c4d`, and `.blend`.

This project uses exclusively imported 3D models for NPCs and general environmental models. These models are created into prefabs which can be instantiated at runtime. These models can be seen in Figure 7.11 The way the NPCs are instantiated into the virtual environment is by putting objects into the environment as spawn points, inserting these spawn points into a list and inserting the prefabs for each animated model into another list. The list of spawn points is then filled with randomly instantiated prefabs from the list of prefabs. This means that the prefab in each spawn point is selected at random. Virtual environment models are not dynamically spawned.

Figure 7.11: The available six skeletonized NPCs

7.2.5.2 Animations

3D animations, as well as 2D animations, are notoriously difficult to implement and incredibly time consuming. Unity has a built-in animator and animation plugin; however, I wouldn't expect most developers to use them for anything too complex.

The animator allows for a series of animation clips to transition between each other in stages. These stages are directed into one another and allow standard parameters and output to be declared, like they are functions. The animator also allows for sub-systems to be declared that can hold their own animation clips and can be directed into and out of from the main system. Creating animation clips themselves in Unity is generally a pretty tedious process even for the most basic animations. Even if you have a skeletonized humanoid model, Unity's animator requires that each object and each property of that object have been defined explicitly in the property view. The user then has to make key frames for each change that needs to happen at the necessary intervals. All of this is a very lengthy and tedious process, so most developers will either use a third-party's animation software, like Blender's animation and rigging [25], or assets from Unity's asset store, like UMotion [50]. These help with animation considerably by skipping the part in Unity where the user has to define specific objects and properties. Due to the skeletonized nature of the models, specific

aspects of the models get pre-defined. None of these tools will get rid of keyframe-ing your way through animations, but some do help this process by allowing the models to automatically move from one keyframe to the next in the most linear way possible.

Small animations, things like general movements that don't require too many moving parts, can be easily done in Unity's animation window. Unity's animation creation also has support for recording animations. This feature is particularly useful if the user is either rotating, scaling, or moving a single object. To use this feature a user simply has to press the record button and then rotate, scale, or move the object. The user must then keyframe at desired time intervals and Unity will linearly change the expected parameters to result in each keyframe at the selected time intervals. For example, if the user were to move an object from coordinates 0,0,0 to 0,10,0 and select 0 and 10 seconds as key frames for each respective position, the object would move one unit in the y direction per second until ten seconds elapse and the object is at the desired coordinates. The user also has the option of importing animations into the Unity asset list. These animations can be found on the asset store or found in third-party websites. As long as the user imports the animation clips into the animator, they will generally work with any humanoid and skeletonized models.

This part of the project utilizes the animator, animation creator, and third party animations to create a list of animations for each and every NPC to perform. These animations are: waving, clapping, dancing, and giving the thumbs up to a user who is doing well in the typing racer. The reason each model can use the same animations is due to the fact that each and every prefab NPC has the same animator controller, which has every animation set as a state that the NPC can use at run-time. It is also important to note that every model needs to be skeletonized in order for the animations to be compatible. These animation states are ran by utilizing a script that takes in a developer created "likelihood of animation". This variable allows animations to be ran more or less frequently depending on developer preference.

7.3 Ambiance

7.3.1 Sound

Background Music (BGM) is one of the smaller details that make huge differences in video games and entertainment. In movies and TV shows, BGM is used to convey emotions and often to incredible impact. This application utilizes background music simply to entice the user to keep playing the game. While it would be outside the scope of this project to include every genre of music or create a system for users to upload their own music to use, having a diversity of music is still somewhat important. Not only does this project associate BGM with a VE, causing the environment to change depending on which music choice they have, but it is also important to allow people who have different tastes in music to enjoy the game and want to be inside it for longer. The genres that are included in the game are: pop, country, and indie. The definitions for each type of music is very vague, and users could argue that one track or another is not of the genre it's associated with. The genre associated with each song was taken from the website in which it was downloaded. Each of these songs had a few other requirements to be chosen for this project. Each song had to have no vocals, as it was determined that vocals could be distracting and generally most games do not have vocals. Each song needed to be categorized in one of the three genres determined above. Each song needed to have a beats per minute of at least eighty to keep the user interested. Slower songs generally give off a more sad feeling and give users low energy. Each song also needed to be copyright and royalty free. While it is not intended to actually sell this game, it is important to not only credit the artists of the music it uses, but to also comply with copyright law. Each genre of music also needed to have at least ten minutes of audio in it. More is better in this instance, so the music does not repeat, but ten minutes was deemed sufficient.

Sound effects (SFX) can also make a large difference for the feel of a game. The more realistic SFX that are tied to realistic actions, the better the user experience and the more likely it is that the user will continue to play. This project uses a key

press sound whenever the user presses a key. Each non player character (NPC) has an audio source component that can play different audio clips when necessary, like a clapping sound when they clap, a cheering sound when they cheer, etc. The audio source component allows for dynamic audio clip playing. This attached to the NPCs allow localized sound, so it doesn't distract from the BGM or other SFX.

7.3.2 Virtual Environment

The virtual environment for this game is really important. The goal of this addition and adaption of typing racers to VR is to showcase the typing methods that are possible in VR and exercise some of the strengths that VR has. One strength is in the ability to feel immersed in a virtual environment, thus the environment matters quite a bit. In the creation of the virtual environments, it was important to make them feel more immersive and make the users want to be in the environments. A good way to do this is by giving the user a choice in what virtual environment they want to participate in. In general, giving users choices can help build user experience, though there is a point where too many options diminishes user experience.

The way that this work gives this option to users is by allowing them to choose not only the type/genre of BGM, but also by associating that music with an environment. For example: Pop music places the user on a stage with a crowd of NPCs watching and emoting. This stage has particle systems that emit particles to look like smoke/fog machines, speakers that don't actually play sound, lasers that change color randomly and fire for a random duration averaging two seconds, and rotating/moving stage lights in a dark environment. See Figure 7.12 for different perspectives in this environment; Country music places the user in a field with corn growing in front of a barn with a bright sun overhead and rolling hills in the background with windmills atop them. See Figure 7.13 for different perspectives in this environment; Indie music has a similar setup to the stage for pop music, but with less NPCs, a smaller room, and some couches in the room. See Figure 7.14 for different perspectives in this environment.

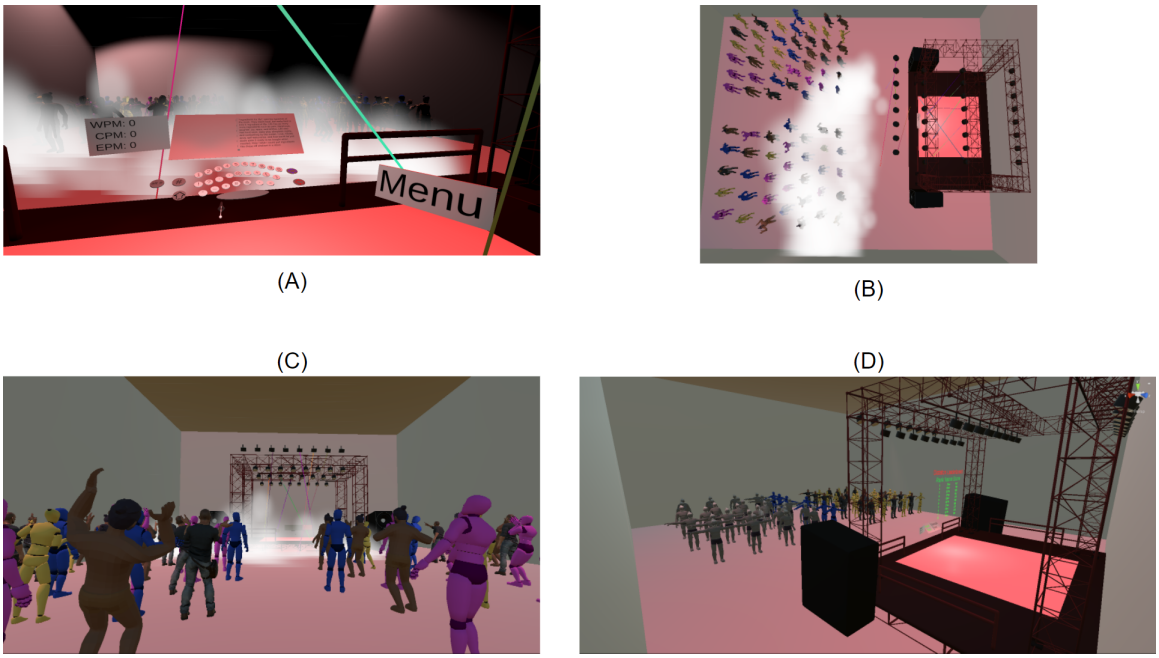


Figure 7.12: The stage environment. (A): The in-game view. (B): The view from the top of the environment facing down. (C): The back of the environment looking towards the user. (D) The view of the environment outside of play, without effects

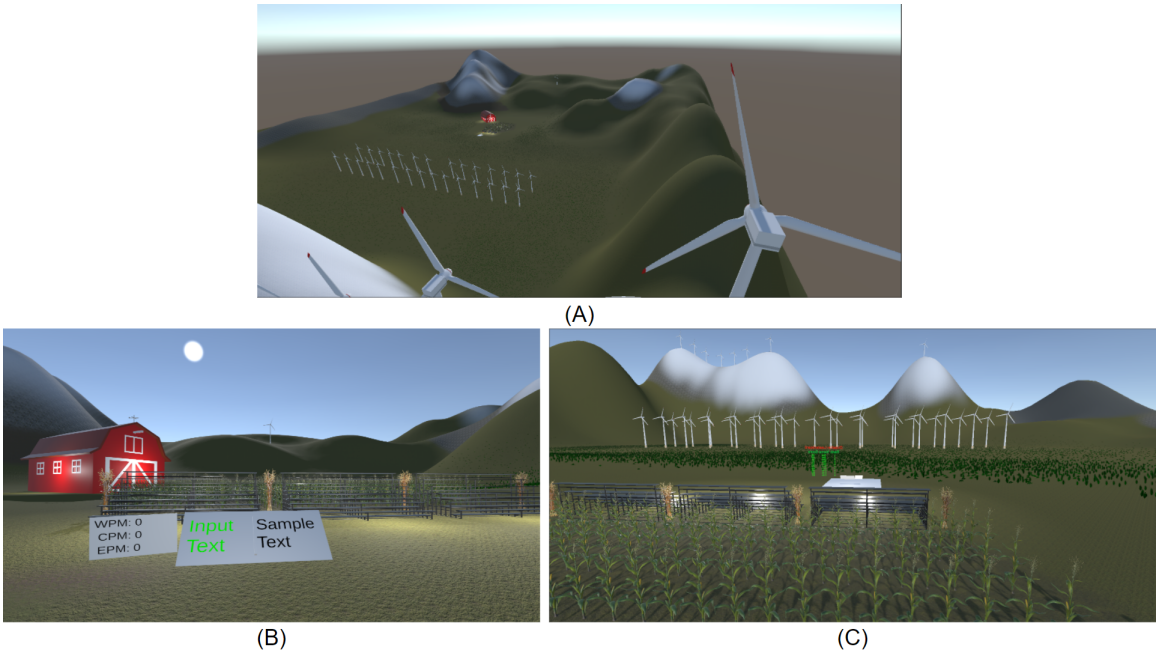


Figure 7.13: The Field environment. (A): The full environment taken from the top of a hill. (B): The in-game viewpoint. (C): The rear view of the environment with the play area in view

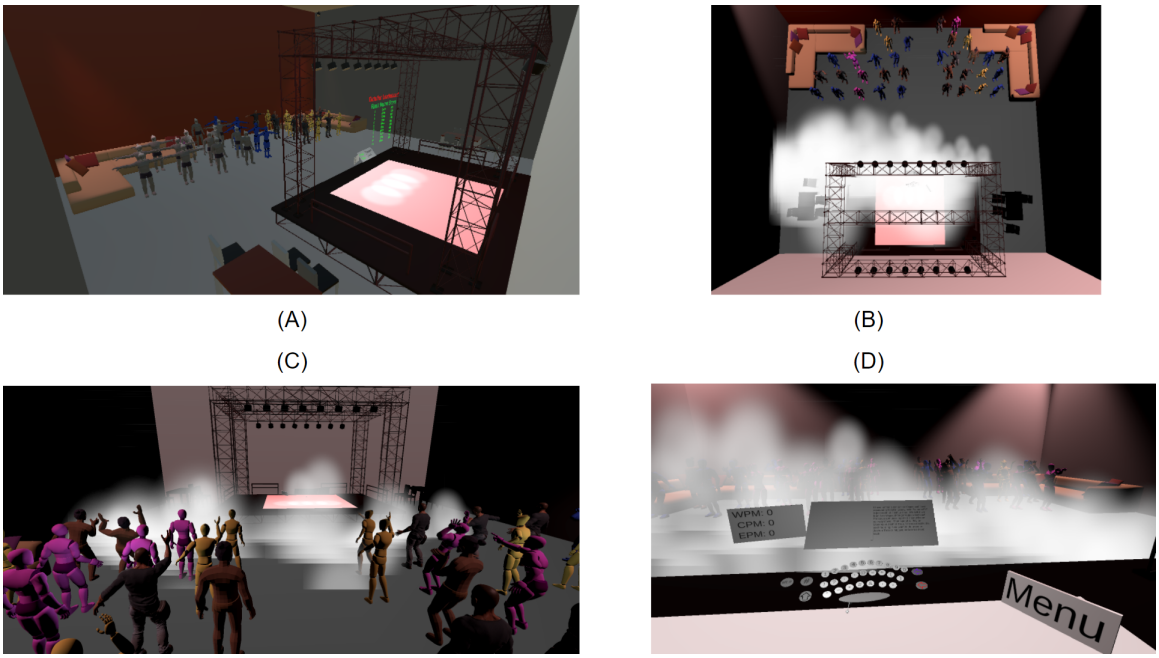


Figure 7.14: The indie stage environment. (A): The view of the environment outside of play, without effects. (B): The top-down view of the environment. (C): The back of the environment looking towards the user. (D): The in-game view.

7.3.3 Lighting

Lighting itself is very difficult to get exactly how a developer might like it. There are difficulties between baking light in directly to the scene, a technique meant for stationary light so it doesn't need to be calculated at run-time, and dynamic area-based lighting, which is a more computationally complex operation with the ability to dynamically change based off of changing light sources and occlusions. Lighting reflections are also quite difficult to calculate and the dispersion of the light may also require non-standard shaders to act properly. Light acts very differently to many different types of materials and many difficulties can arise due to that fact.

Raytracing is a technique for modeling light in digital mediums. This technique can simulate many natural phenomena that are due to light like: reflection, refraction, soft shadows, scattering, depth of field, motion blur, caustics, ambient occlusion, and dispersion. This technique can also trace the path of sound waves similarly to that of light waves. Raytracing is not a particularly new technology, it was first created

as a way to draw objects in the sixteenth century in the apparatus named "Dürer's door" [42]. The technology was used mainly in computed-generated images and in film/television VFX, where long render times can be tolerated.

Within the last five years, hardware acceleration for real-time ray tracing has become standard in graphics cards. This has caused video games to increase realism drastically, and they have shown this realism particularly well with light in regards to water. Water effects showcase the raytracing technology particularly well due to the fact that water reflects, refracts, scatters, and disperses light. Water also showcases ambient occlusion and caustics. Unity has made support for real-time ray tracing in a render pipeline, but this has not been fully released yet. Regardless, this is a very exciting technology that is soon to be released and implemented in large platforms, like Unity and Unreal. This project does not use ray-tracing, but the implications of this technology for high-end real-time rendering in VR is quite large and should be discussed when referencing lighting in any VR application. With this technology, VR applications will become even more immersive and realistic, causing future applications in VR to have a greater impact.

The lighting in this application is quite simple. Depending on the environment chosen by the user, lighting is given by either a directional light, spotlight, or point light. Directional lights are analogous to the sun in a scene. The developer can rotate this light to cause either baked in light or real-time rendering from a given direction. If this light is rotated parallel to the ground, it will look like sunset or sunrise in terms of where the shadows are coming from, how long they are, and what the skybox looks like. If the light is rotated straight up, everything is dark and the skybox turns black to make it look like nighttime. Spotlights is a light emitter that points in a cone at a given rotation. This light emitter allows both baking and real-time rendering of light. This component can change the intensity of the light, the angle the light is facing, and the spread of the cone of light, making the flat side of the cone change in area. These are used as stage lights or specific highlighting lights. Area lights and point lights work by lighting up a set area. Point lights are simple spheres of light that developers

can determine the range of. Light within the range loses intensity with the inverse square law, resulting in no light at the outer bounds of the range. This light can be both baked or real-time rendered. Area lights have uniform light emitted from one face of the rectangle or disc. This light is required to be baked into lightmaps. Area lights are not used in this project, but the definition is included for completeness. Another source of light that is a little different from the previously discussed light sources are the lasers that are a part of the pop music stage environment. These lasers are drawn with line renderers that start at the stage light models and then end at the floor. These lasers emit random colors for random periods of time. The colors they emit are chosen randomly and emit based off of a gradient from a starting color to an ending color. These lasers are set to emit light based off of the color of the gradients.

7.4 Future Work

Lighting is a large part of any 3D application. Good lighting can cause greater immersion and presence. This application uses basic lighting to properly showcase assets and environments. Despite this, utilizing raytracing would improve this project drastically. Many of the lights used in this project are to help light areas or specific necessary objects due to the very small amount of reflection and scattering of light that happens natively. A more accurate dispersion, reflection, and scattering of light would cut down on the number of additional lights needed in the environments, as well as cut down on the intensity and size of lights. This would improve the realism in the scenes and it would better display the environment to the user.

User avatar and avatar selection would be an interesting addition to this work. Personally selected avatars that can be customized have shown to improve immersion and presence. After running the user study talked about in Chapter 5, users also seem to have a good time selecting their features. They also often went through and selected multiple different combinations of features, just to see how the avatar looked in each. This would also allow multiplayer matches to feel more competitive

as it would look like there is another person beside the user that they are competing against, rather than just a number.

There are many genres of music, this application implements three genres that each have multiple sub-genres. While none of the music in this application would likely be frowned upon by most users, the music selection is severely limited. Most new genres wouldn't necessarily need new environments to be added, but there are certain genres that wouldn't necessarily be played in any of the environments that have been created, so they would seem odd to play in those environments.

Of the identified typing methods in this paper, a few aren't implemented. Namely the dwell or hands-free based typing methods. These typing methods help those with disabilities use VR and generally make applications more accessible. Despite these methods being shown to be slower than the other input methods, they would make a great addition to this application.

Currently, NPC animations are triggered randomly based off of a developer set "likeliness of animation". A good adaptation of this is limiting animations based off of user results. Meaning that animations should be limited when the user is doing poorly, or maybe new animations are displayed that show more negative emotions. The number of animations should also be expanded when the user is performing well. Generally, more indications of user success or failure would help elevate this applications user experience.

A user study would also expand this work. A comparison between all identified or implemented VR typing methods would require a very large user study to be statistically relevant. This user study would seek to understand user experience and usability between input methods, as well as any correlation between the two and the speed or error rate of the participants.

References

[1] Frederick Aardema, Kieron O'Connor, Sophie Côté, and Annie Taillon. Virtual reality induces dissociation and lowers sense of presence in objective reality. In volume 13 of number 4, pages 429–435, 2010. DOI: 10.1089/cyber.2009.0164. URL: https://doi.org/10.1089/cyber.2009.0164. PMID: 20712501.

[2] Justin M. Albani and David I. Lee. Virtual reality-assisted robotic surgery simulation. *Journal of Endourology*, 21:285–287, March 2007. DOI: 10.1089/end.2007.9978.

[3] Amazon. Notwirez - twitch. URL: https://www.twitch.tv/notwirez (visited on 06/03/0022).

[4] Amazon. Twitch. URL: https://www.twitch.tv/ (visited on 06/03/0022).

[5] Steven J. Anbro, Alison J. Szarko, Ramona A. Houmanfar, Amber M. Maraccini, Laura H. Crosswell, Frederick C. Harris, Michelle Rebaleati, and Luka Starmer. Using virtual simulations to assess situational awareness and communication in medical and nursing education: a technical feasibility study. *Journal of Organizational Behavior Management*, 40(1-2):129–139, April 2020. DOI: 10.1080/01608061.2020.1746474. URL: https://doi.org/10.1080/01608061.2020.1746474.

[6] Steven J. Anbro, Alison J. Szarko, Ramona A. Houmanfar, Amber M. Maraccini, Laura H. Crosswell, Frederick C. Harris, Michelle Rebaleati, and Luka Starmer. Using virtual simulations to assess situational awareness and communication in medical and nursing education: a technical feasibility study. *Journal of Organizational Behavior Management*, 40(1-2):129–139, April 2020. DOI: 10.1080/01608061.2020.1746474. URL: https://doi.org/10.1080/01608061.2020.1746474.

[7] Mahdi Azmandian, Timofey Grechkin, and Evan Suma Rosenberg. An evaluation of strategies for two-user redirected walking in shared physical spaces. In *2017 IEEE Virtual Reality (VR)*, pages 91–98, 2017. DOI: 10.1109/VR.2017.7892235.

[8] D. Baur, C. Pfeifle, and C. E. Heyde. Cervical spine injury after virtual reality gaming: a case report. *Journal of Medical Case Reports*, 15(1), May 2021. DOI: 10.1186/s13256-021-02880-9. URL: https://doi.org/10.1186/s13256-021-02880-9.

[9] Alim Louis Benabid, Thomas Costecalde, Andrey Eliseyev, Guillaume Charvet, Alexandre Verney, Serpil Karakas, Michael Foerster, Aurélien Lambert, Boris Morinière, Neil Abroug, Marie-Caroline Schaeffer, Alexandre Moly, Fabien Sauter-Starace, David Ratel, Cecile Moro, Napoleon Torres-Martinez, Lilia Langar, Manuela Oddoux, Mircea Polosan, Stephane Pezzani, Vincent Auboiroux, Tetiana Aksenova, Corinne Mestais, and Stephan Chabardes. An exoskeleton controlled by an epidural wireless brain–machine interface in a tetraplegic patient: a proof-of-concept demonstration. *The Lancet Neurology*, 18(12):1112–1122, 2019. ISSN: 1474-4422. DOI: https://doi.org/10.1016/S1474-4422(19)30321-7. URL: https://www.sciencedirect.com/science/article/pii/S1474442219303217.

[10] Costas Boletsis and Stian Kongsvik. Controller-based text-input techniques for virtual reality: an empirical comparison. *International Journal of Virtual Reality*, 19(3):2–15, August 2019. DOI: 10.20870/IJVR.2019.19.3.2917. URL: https://ijvr.eu/article/view/2917.

[11] Costas Boletsis and Stian Kongsvik. Text input in virtual reality: a preliminary evaluation of the drum-like vr keyboard. *Technologies*, 7(2), 2019. ISSN: 2227-7080. DOI: 10.3390/technologies7020031. URL: https://www.mdpi.com/2227-7080/7/2/31.

[12] Cave2: next-generation virtual-reality and visualization hybrid environment for immersive simulation and information analysis, April 2009. URL: https://www.evl.uic.edu/research/2016 (visited on 06/03/0022).

[13] Moohyun Cha, Soonhung Han, Jaikyung Lee, and Byungil Choi. A virtual reality based fire training simulator integrated with fire dynamics data. *Fire Safety Journal*, 50:12–24, 2012. ISSN: 0379-7112. DOI: https://doi.org/10.1016/j.firesaf.2012.01.004. URL: https://www.sciencedirect.com/science/article/pii/S0379711212000136.

[14] Munmun De Choudhury, Emre Kiciman, Mark Dredze, Glen Coppersmith, and Mrinal Kumar. Discovering shifts to suicidal ideation from mental health content in social media. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, May 2016. DOI: 10.1145/2858036.2858207. URL: https://doi.org/10.1145/2858036.2858207.

[15] Kelly B. Clancy and Thomas D. Mrsic-Flogel. The sensory representation of causally controlled objects. *Neuron*, 109(4):677–689.e4, February 2021. DOI: 10.1016/j.neuron.2020.12.001. URL: https://doi.org/10.1016/j.neuron.2020.12.001.

[16] Viviane Clay, Peter König, and Sabine U. König. Eye tracking in virtual reality. *Journal of Eye Movement Research*, 12(1), April 2019. DOI: 10.16910/jemr.12.1.3. URL: https://doi.org/10.16910/jemr.12.1.3. https://doi.org/10.16910/jemr.12.1.3.

[17] Justice Steven Colby. *American Sign Language Gesture Recognition using Motion Tracking Gloves in VR*. Master's thesis, University of Nevada, Reno, Reno, NV 89557, 2022. https://www.cse.unr.edu/~fredh/papers/thesis/083-colby/thesis.pdf (Last accessed: 6/22/2022).

[18] Timothy R. Coles, Dwight Meglan, and Nigel W. John. The role of haptics in medical training simulators: a survey of the state of the art. *IEEE Transactions on Haptics*, 4(1):51–66, 2011. DOI: 10.1109/TOH.2010.19.

[19] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques - SIGGRAPH '93*. ACM Press, 1993. DOI: 10.1145/166117.166134. URL: https://doi.org/10.1145/166117.166134.

[20] Matthew M. Davis, Joseph L. Gabbard, Doug A. Bowman, and Dennis Gracanin. Depth-based 3d gesture multi-level radial menu for virtual object manipulation. In *2016 IEEE Virtual Reality (VR)*, pages 169–170, 2016. DOI: 10.1109/VR.2016.7504707.

[21] Jing Du, Yangming Shi, Chao Mei, John Quarles, and Wei Yan. Communication by interaction: a multiplayer vr environment for building walkthroughs. In *Construction Research Congress 2016*, pages 2281–2290, 2016. DOI: 10.1061/9780784479827.227. eprint: https://ascelibrary.org/doi/pdf/10.1061/9780784479827.227. URL: https://ascelibrary.org/doi/abs/10.1061/9780784479827.227.

[22] Digital Extremes. Warframe: ninjas play free. URL: https://www.warframe.com/landing (visited on 06/03/0022).

[23] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. Fixation prediction for 360° video streaming in head-mounted virtual reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, June 2017. DOI: 10.1145/3083165.3083180. URL: https://doi.org/10.1145/3083165.3083180.

[24] Mark A. Finney, Isaac C. Grenfell, Charles W. McHugh, Robert C. Seli, Diane Trethewey, Richard D. Stratton, and Stuart Brittain. A method for ensemble wildland fire simulation. *Environmental Modeling & Assessment*, 16(2):153–167, November 2010. DOI: 10.1007/s10666-010-9241-3. URL: https://doi.org/10.1007/s10666-010-9241-3.

[25] Blender Foundation. Animation & rigging. URL: https://www.blender.org/features/animation/ (visited on 06/03/0022).

[26] Blender Foundation. Free and open 3d creations. URL: https://www.blender.org (visited on 06/03/0022).

[27] Anthony G. Gallagher, E Matt Ritter, Howard Champion, Gerald Higgins, Marvin P. Fried, Gerald Moses, C Daniel Smith, and Richard M. Satava. Virtual reality simulation for the operating room. 241(2):364–372, 2005. DOI: 10.1097/01.sla.0000151982.85062.80.

[28] Beat Games. Beat saber - vr rhythm game. URL: https://beatsaber.com/ (visited on 06/03/0022).

[29] Andy Mario Garcia. *An Advanced Wildfire Simulator: vFirelib.v2*. Master's thesis, University of Nevada, Reno, Reno, NV 89557, 2018. https://www.cse.unr.edu/~fredh/papers/thesis/074-garcia/thesis.pdf (Last accessed: 5/2/2019).

[30] Google. Angular. URL: https://angular.io/ (visited on 05/31/0022).

[31] Google. Firebase. URL: https://www.firebase.google.com/ (visited on 06/03/0022).

[32] Google. Google earth VR. URL: https://arvr.google.com/earth/ (visited on 04/28/2022).

[33] John Paulin Hansen, Anders Sewerin Johansen, Dan Witzner Hansen, Kenji Ito, and Satoru Mashino. Command without a click: dwell time typing by mouse and gaze selections. In *Interact*, volume 3, pages 121–128. Citeseer, 2003.

[34] Hong S. He and David J. Mladenoff. Spatially explicit and stochastic simulation of forest-landscape fire disturbance and succession. *Ecology*, 80(1):81–99, 1999. DOI: https://doi.org/10.1890/0012-9658(1999)080[0081:SEASSO]2.0.CO;2. eprint: https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.1890/0012-9658\%281999%29080%5B0081%3ASEASSO%5D2.0.CO%3B2. URL: https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/0012-9658\%281999\%29080\%5B0081\%3ASEASSO\%5D2.0.CO\%3B2.

[35] Patient Safety Quality Healthcare. Patient handoffs: the gap where mistakes are made. *Patient Safety Monitor Journal*, November 2017. URL: http://www.hcpro.com/QPS-330353-234/Patient-handoffs-The-gap-where-mistakes-are-made.html?sessionGUID=edbe08b4-46e3-0e62-79c3-b0d7966ab079\&webSyncID=3298c37b-1099-3ae8-b921-25094b55dc8c (visited on 12/01/2017). Last accessed 6 October 2021.

[36] Morton L. Heilig. Sensorama simulator. Patent (US3050870A). August 1962. URL: https://patents.google.com/patent/US3050870A/en (visited on 05/31/2022).

[37] Morton L. Heilig. Stereoscopic-television apparatus for individual use. Patent (US2955156A). October 1960. URL: https://patents.google.com/patent/US2955156A/en (visited on 05/31/2022).

[38] Steven Hickson, Nick Dufour, Avneesh Sud, Vivek Kwatra, and Irfan Essa. Eyemotion: classifying facial expressions in VR using eye-tracking cameras. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, January 2019. DOI: 10.1109/wacv.2019.00178. URL: https://doi.org/10.1109/wacv.2019.00178.

[39] Roger V. Hoang, Joseph D. Mahsman, David T. Brown, Michael A. Penick, Frederick C. Harris, and Timothy J. Brown. Vfire: virtual fire in realistic environments. In *2008 IEEE Virtual Reality Conference*, pages 261–262, 2008. DOI: 10.1109/VR.2008.4480791.

[40] Roger V. Hoang, Joseph D. Mahsman, David T. Brown, Michael A. Penick, Frederick C. Harris Jr., and Timothy J. Brown. Vfire: virtual fire in realistic environments. In *2008 IEEE Virtual Reality Conference*, pages 261–262, 2008. DOI: 10.1109/VR.2008.4480791.

[41] Roger V Hoang, Matthew R Sgambati, Timothy J Brown, Daniel S Coming, and Frederick C Harris Jr. Vfire: immersive wildfire simulation and visualization. *Computers & Graphics*, 34(6):655–664, 2010.

[42] Georg Rainer Hofmann. Who invented ray tracing? *The Visual Computer*, 6(3):120–124, May 1990. DOI: 10.1007/bf01911003. URL: https://doi.org/10.1007/bf01911003.

[43] HTC. Steamvr unity plugin. [online]. URL: https://valvesoftware.github.io/steamvr_unity_plugin/ (visited on 11/28/2021).

[44] HTC Corporation. Documentation - Developer Resources: SRanipal Unity SDK for the HTC Vive Pro Eye. [online]. URL: https://developer-express.vive.com/resources/vive-sense/eye-and-facial-tracking-sdk/documentation/ (visited on 11/28/2021).

[45] HTC Corporation. VIVE Pro Eye Overview: VIVE United States. [online]. URL: https://www.vive.com/us/product/vive-pro-eye/overview/ (visited on 11/30/2021).

[46] HTC Corporation. VIVE United States: Next-level VR Headsets and Apps. [online]. URL: https://www.vive.com/us/ (visited on 11/27/2021).

[47] I-Illusions. Space pirate trainer. URL: https://www.spacepiratetrainer.com/ (visited on 05/31/0022).

[48] Radhwan Hussein Ibrahim and Dhiaa-Alrahman Hussein. Assessment of visual, auditory, and kinesthetic learning style among undergraduate nursing students. *International Journal of Advanced Nursing Studies*, 5(1):1–4, December 2015. DOI: 10.14419/ijans.v5i1.5124. URL: https://doi.org/10.14419/ijans.v5i1.5124.

[49] VRChat inc. VRChat Hompage. URL: https://hello.vrchat.com/ (visited on 05/31/0022).

[50] Soxware Interactive. URL: https://assetstore.unity.com/packages/tools/animation/umotion-community-animation-editor-95986#description (visited on 06/03/0022).

[51] JetBrains. Riderflow for unity: scenery tool to build and manage your 3d space. URL: https://www.jetbrains.com/riderflow/ (visited on 05/31/0022).

[52] R. S. Kennedy, M. G. Lilienthal, K. S. Berbaum, D. R. Baltzley, and M. E. McCauley. Simulator sickness in U.S. Navy flight simulators. In volume 60 of number 1, pages 10–16, January 1989.

[53] Barrett T. Kitch, Jeffrey B. Cooper, Warren M. Zapol, Matthew M. Hutter, Jessica Marder, Andrew Karson, and Eric G. Campbell. Handoffs causing patient harm: a survey of medical and surgical house staff. *The Joint Commission Journal on Quality and Patient Safety*, 34(10):563–570d, 2008. ISSN: 1553-7250. DOI: https://doi.org/10.1016/S1553-7250(08)34071-9. URL: https://www.sciencedirect.com/science/article/pii/S1553725008340719.

[54] Simon Kloiber, Volker Settgast, Christoph Schinko, Martin Weinzerl, Johannes Fritz, Tobias Schreck, and Reinhold Preiner. Immersive analysis of user motion in VR applications. *The Visual Computer*, 36(10-12):1937–1949, August 2020. DOI: 10.1007/s00371-020-01942-1. URL: https://doi.org/10.1007/s00371-020-01942-1.

[55] Dong-Yong Lee, Yong-Hun Cho, and In-Kwon Lee. Real-time optimal planning for redirected walking using deep q-learning. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 63–71, 2019. DOI: 10.1109/VR.2019.8798121.

[56] Christopher Lewis, Sven Diaz-Juarez, Steven Anbro, Alison Szarko, Ramona Houmanfar, Laura Crosswell, Michelle Rebaleati, Luka Starmer, and Frederick Harris. An application for simulating patient handoff using 360 video and eye tracking in virtual reality. In Bidyut Gupta, Ajay Bandi, and Mohammad Hossain, editors, *Proceedings of 37th International Conference on Computers and Their Applications*, volume 82 of *EPiC Series in Computing*, pages 141–149. EasyChair, 2022. DOI: 10.29007/82j6. URL: https://easychair.org/publications/paper/ztdK.

[57] Christopher Lewis, Ronn Siedrik Quijada, and Frederick C. Harris. vFireVI: 3d virtual interface for vFire. In *Advances in Intelligent Systems and Computing*, pages 309–315. Springer International Publishing, 2020. DOI: 10.1007/978-3-030-43020-7_41. URL: https://doi.org/10.1007/978-3-030-43020-7_41.

[58] Ning-Han Liu, Cheng-Yu Chiang, and Hsuan-Chin Chu. Recognizing the degree of human attention using eeg signals from mobile sensors. *Sensors*, 13(8):10273–10286, 2013. ISSN: 1424-8220. DOI: 10.3390/s130810273. URL: https://www.mdpi.com/1424-8220/13/8/10273.

[59] Wen-Chih Lo, Ching-Ling Fan, Jean Lee, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 360° video viewing dataset in head-mounted virtual reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, June 2017. DOI: 10.1145/3083187.3083219. URL: https://doi.org/10.1145/3083187.3083219.

[60] Cristian Luciano, Pat Banerjee, and Thomas DeFanti. Haptics-based virtual reality periodontal training simulator. *Virtual Reality*, 13(2):69–85, February 2009. DOI: 10.1007/s10055-009-0112-7. URL: https://doi.org/10.1007/s10055-009-0112-7.

[61] Ewa Luger and Abigail Sellen. "Like having a really bad PA": The gulf between user expectation and experience of conversational agents. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, California, USA*, CHI '16, 5286–5297, New York, NY, USA. Association for Computing Machinery, 2016. ISBN: 9781450333627. DOI: 10.1145/2858036.2858288. URL: https://doi.org/10.1145/2858036.2858288.

[62] I Scott MacKenzie and Kumiko Tanaka-Ishii. *Text entry systems*. Morgan Kaufmann, Oxford, England, March 2007. ISBN: 978-0123735911.

[63] Päivi Majaranta, Ulla-Kaija Ahola, and Oleg Špakov. Fast gaze typing with an adjustable dwell time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, USA*, CHI '09, 357–360, New York, NY, USA. Association for Computing Machinery, 2009. ISBN: 9781605582467. DOI: 10.1145/1518701.1518758. URL: https://doi.org/10.1145/1518701.1518758.

[64] Manifesto for agile software development. URL: http://agilemanifesto.org/ (visited on 06/03/0022).

[65] Microsoft. Ide and code editor for software developers and teams, May 2022. URL: https://visualstudio.microsoft.com/ (visited on 06/03/0022).

[66] Mark Roman Miller, Fernanda Herrera, Hanseul Jun, James A. Landay, and Jeremy N. Bailenson. Personal identifiability of user tracking data during observation of 360-degree VR video. In volume 10 of number 1. Springer Science and Business Media LLC, October 2020. DOI: 10.1038/s41598-020-74486-y. URL: https://doi.org/10.1038/s41598-020-74486-y.

[67] Mirantis. Home - Docker. URL: https://www.docker.com/ (visited on 06/03/0022).

[68] Martez E. Mott, Shane Williams, Jacob O. Wobbrock, and Meredith Ringel Morris. Improving dwell-based gaze typing with dynamic, cascading dwell times. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, Colorado, USA*, CHI '17, 2558–2570, New York, NY, USA. Association for Computing Machinery, 2017. ISBN: 9781450346559. DOI: 10.1145/3025453.3025517. URL: https://doi.org/10.1145/3025453.3025517.

[69] Fares Moustafa and Anthony Steed. A longitudinal study of small group interaction in social virtual reality. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*. ACM, November 2018. DOI: 10.1145/3281505.3281527. URL: https://doi.org/10.1145/3281505.3281527.

[70] Martin Müller, Jonas Jürgens, Marcus Redaèlli, Karsten Klingberg, Wolf E Hautz, and Stephanie Stock. Impact of the communication and patient handoff tool sbar on patient safety: a systematic review. *BMJ Open*, 8(8), 2018. ISSN: 2044-6055. DOI: 10.1136/bmjopen-2018-022202. eprint: htctps://bmjopen.bmj.com/content/8/8/e022202.full.pdf. URL: https://bmjopen.bmj.com/content/8/8/e022202.

[71] Mahdi Nabiyouni, Ayshwarya Saktheeswaran, Doug A. Bowman, and Ambika Karanth. Comparing the performance of natural, semi-natural, and non-natural locomotion techniques in virtual reality. In *2015 IEEE Virtual Reality (VR)*, pages 243–244, 2015. DOI: 10.1109/VR.2015.7223386.

[72] NASA. GVIS - GRUVE Lab - glenn research center. URL: https://www1.grc.nasa.gov/facilities/gvis/gruve-lab/ (visited on 05/31/0022).

[73] Don Norman. *The design of everyday things*. en. Basic Books, London, England, 2nd edition, November 2013. ISBN: 978-0465050659.

[74] Don Norman. *The psychology of everyday things*. Basic Books, London, England, May 1988. ISBN: 978-0465067091.

[75] Oculus — vr headsets, games, & equipment. URL: https://www.oculus.com/ (visited on 10/08/2021).

[76] Ilesanmi Olade, Charles Fleming, and Hai-Ning Liang. BioMove: biometric user identification from human kinesiological movements for virtual reality systems. In volume 20 of number 10, page 2944. MDPI AG, May 2020. DOI: 10.3390/s20102944. URL: https://doi.org/10.3390/s20102944.

[77] ORamaVR. ORamaVR - the world's most intelligent VR medical training simulations. [online]. URL: https://oramavr.com/ (visited on 10/15/2021).

[78] Veronica S Pantelidis. Reasons to use virtual reality in education and training courses and a model to determine when to use virtual reality. *Themes in Science and Technology Education*, 2(1-2):59–70, 2010.

[79] George D. Park, R. Wade Allen, Dary Fiorentino, Theodore J. Rosenthal, and Marcia L. Cook. Simulator sickness scores according to symptom susceptibility, age, and gender for an older driver assessment study. In volume 50 of number 26, pages 2702–2706. SAGE Publications, October 2006. DOI: 10.1177/154193120605002607. URL: https://doi.org/10.1177/154193120605002607.

[80] Ken Pfeuffer, Matthias J. Geiger, Sarah Prange, Lukas Mecke, Daniel Buschek, and Florian Alt. Behavioural biometrics in VR. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, May 2019. DOI: 10.1145/3290605.3300340. URL: https://doi.org/10.1145/3290605.3300340.

[81] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. ACM, October 2016. DOI: 10.1145/2980055.2980056. URL: https://doi.org/10.1145/2980055.2980056.

[82] Kazi Shahzabeen Rahnuma, Abdul Wahab, Norhaslinda Kamaruddin, and Hariyati Majid. Eeg analysis for understanding stress based on affective model basis function. In *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*, pages 592–597, 2011. DOI: 10.1109/ISCE.2011.5973899.

[83] Sharif Razzaque, Zachariah Kohn, and Mary C. Whitton. Redirected Walking. In *Eurographics 2001 - Short Presentations*. Eurographics Association, 2001. DOI: 10.2312/egs.20011036.

[84] Anke Verena Reinschluessel, Joern Teuber, Marc Herrlich, Jeffrey Bissel, Melanie van Eikeren, Johannes Ganser, Felicia Koeller, Fenja Kollasch, Thomas Mildner, Luca Raimondo, Lars Reisig, Marc Ruedel, Danny Thieme, Tobias Vahl, Gabriel Zachmann, and Rainer Malaka. Virtual reality for user-centered design and evaluation of touch-free interaction techniques for navigating medical images in the operating room. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, May 2017. DOI: 10.1145/3027063.3053173. URL: https://doi.org/10.1145/3027063.3053173.

[85] RockVR video capture. URL: https://assetstore.unity.com/packages/tools/video/video-capture-75653 (visited on 10/08/2021).

[86] Robert O. Roswell, Courtney D. Cogburn, Jack Tocco, Johanna Martinez, Catherine Bangeranye, Jeremy N. Bailenson, Michael Wright, Jennifer H. Mieres, and Lawrence Smith. Cultivating empathy through virtual reality: advancing conversations about racism, inequity, and climate in medicine. In volume 95 of number 12, pages 1882–1886. Ovid Technologies (Wolters Kluwer Health), July 2020. DOI: 10.1097/acm.0000000000003615. URL: https://doi.org/10.1097/acm.0000000000003615.

[87] G. S. Ruthenbeck and K. J. Reynolds. Virtual reality for medical training: the state-of-the-art. *Journal of Simulation*, 9(1):16–26, February 2015. DOI: 10.1057/jos.2014.14. URL: https://doi.org/10.1057/jos.2014.14.

[88] Dimitrios Saredakis, Ancret Szpak, Brandon Birckhead, Hannah A Keage, Albert Rizzo, and Tobias Loetscher. Factors associated with virtual reality sickness in head-mounted displays: a systematic review and meta-analysis, December 2019. DOI: 10.3389/fnhum.2020.00096. URL: psyarxiv.com/7u4hn.

[89] Shohel Sayeed, Nidal S. Kamel, and Rosli Besar. Virtual reality based dynamic signature verification using data glove. In *2007 International Conference on Intelligent and Advanced Systems*, pages 1260–1264, 2007. DOI: 10.1109/ICIAS.2007.4658586.

[90] Bill N. Schilit, N. Adams, and Roy Want. Context-aware computing applications. In *1994 First Workshop on Mobile Computing Systems and Applications*, pages 85–90, 1994. DOI: 10.1109/WMCSA.1994.16.

[91] Neal E. Seymour, Anthony G. Gallagher, Sanziana A. Roman, Michael K. O'Brien, Vipin K. Bansal, Dana K. Andersen, and Richard M. Satava. Virtual reality training improves operating room performance. In volume 236 of number 4, pages 458–464. Ovid Technologies (Wolters Kluwer Health), October 2002. DOI: 10.1097/00000658-200210000-00008. URL: https://doi.org/10.1097/00000658-200210000-00008.

[92] Jessica Smith, Lee Barfed, Sergiu M. Dasclu, and Frederick C. Harris. Highly parallel implementation of forest fire propagation models on the gpu. In *2016 International Conference on High Performance Computing & Simulation (HPCS)*, pages 917–924, 2016. DOI: 10.1109/HPCSim.2016.7568432.

[93] Jessica Elizabeth Smith. *vFireLib: A Forest Fire Simulation Library Implemented on the GPU*. Master's thesis, University of Nevada, Reno, Reno, NV 89557, 2016. https://www.cse.unr.edu/~fredh/papers/thesis/064-smith/thesis.pdf (Last accessed: 5/2/2019).

[94] Jeongmin Son, Sunggeun Ahn, Sunbum Kim, and Geehyuk Lee. Improving two-thumb touchpad typing in virtual reality. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems, Glasgow, Scotland UK*, CHI EA '19, 1–6, New York, NY, USA. Association for Computing Machinery, 2019. ISBN: 9781450359719. DOI: 10.1145/3290607.3312926. URL: https://doi.org/10.1145/3290607.3312926.

[95] Yale Song, David Demirdjian, and Randall Davis. Continuous body and hand gesture recognition for natural human-computer interaction. *ACM Trans. Interact. Intell. Syst.*, 2(1), March 2012. ISSN: 2160-6455. DOI: 10.1145/2133366.2133371. URL: https://doi.org/10.1145/2133366.2133371.

[96] Misha Sra. Asymmetric design approach and collision avoidance techniques for room-scale multiplayer virtual reality. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology, Tokyo, Japan*, UIST '16 Adjunct, 29–32, New York, NY, USA. Association for Computing Ma-

chinery, 2016. ISBN: 9781450345316. DOI: 10.1145/2984751.2984788. URL: https://doi.org/10.1145/2984751.2984788.

[97] Ivan E. Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*. ACM Press, 1968. DOI: 10.1145/1476589.1476686. URL: https://doi.org/10.1145/1476589.1476686.

[98] Tabnine. Code faster with ai code completions — Tabnine. URL: https://www.tabnine.com/ (visited on 06/03/0022).

[99] thetuvix, keveleigh, and DCtheGeek. Voice input in Unity - Mixed Reality. en-us, May 2021. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/voice-input-in-unity (visited on 04/25/2022).

[100] Jerald Thomas and Evan Suma Rosenberg. A general reactive algorithm for redirected walking using artificial potential functions. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 56–62, 2019. DOI: 10.1109/VR.2019.8797983.

[101] Tobii AB. Providing technologies & solutions for a better world - Tobii. [online]. URL: https://tobii.com/ (visited on 11/27/2021).

[102] Touch of Life Technologies Inc. TolTech - ArthroSim Arthroscopy Simulator. [online]. URL: https://www.toltech.net/medical-simulators/products/arthrosim-arthroscopy-simulator (visited on 10/15/2021).

[103] Steven Tweedie. How do you move in virtual reality? With a treadmill like this one I just tried, January 2015. URL: https://www.businessinsider.com/virtuix-omni-virtual-reality-treadmill-hands-on-2015-1 (visited on 06/03/0022).

[104] Unity Technologies. Unity real-time development platform: 3D, 2D VR & AR engine. [online]. URL: https://unity.com/ (visited on 11/27/2021).

[105] Valve. Half-life: alyx on steam. URL: https://store.steampowered.com/app/546560/HalfLife_Alyx/ (visited on 06/03/0022).

[106] Virtuix. Omni one — the future of gaming. URL: https://omni.virtuix.com/ (visited on 05/31/0022).

[107] Vive facial tracker. URL: https://www.vive.com/us/accessory/facial-tracker/ (visited on 10/08/2021).

[108] VIVE Pro Eye specs. URL: https://www.vive.com/us/product/vive-pro-eye/specs/ (visited on 10/08/2021).

[109] Vive wave — htc vive. URL: https://developer.vive.com/us/wave/ (visited on 10/08/2021).

[110] Voice for pun2. URL: https://doc.photonengine.com/en-US/voice/current/getting-started/voice-for-pun (visited on 10/08/2021).

[111] Vpl-dataglove, 1987. URL: https://www.britannica.com/technology/VPL-DataGlove (visited on 06/03/0022).

[112] Vpl research, April 2021. URL: https://en.wikipedia.org/wiki/VPL_Research (visited on 06/03/0022).

[113] Vr usage & consumer attitudes, wave IV, May 2022. URL: https://artillry.co/artillry-intelligence/vr-usage-consumer-attitudes-wave-vi/ (visited on 06/03/2022).

[114] Michael L. Wilson and Amelia J. Kinsela. Absence of gender differences in actual induced hmd motion sickness vs. pretrial susceptibility ratings. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 61(1):1313–1316, 2017. DOI: 10.1177/1541931213601810. URL: https://doi.org/10.1177/1541931213601810.

[115] Rui Wu, Chao Chen, Sajjad Ahmad, John M. Volk, Cristina Luca, Frederick C. Harris, Jr., and Sergiu M. Dascalu. A real-time web-based wildfire simulation system:4964–4969, 2016. DOI: 10.1109/IECON.2016.7793478.

[116] Ying Wu and Thomas S. Huang. Vision-based gesture recognition: a review. In Annelies Braffort, Rachid Gherbi, Sylvie Gibet, Daniel Teil, and James Richardson, editors, *Gesture-Based Communication in Human-Computer Interaction*, pages 103–115, Berlin, Heidelberg. Springer Berlin Heidelberg, 1999. ISBN: 978-3-540-46616-1. DOI: 10.1007/3-540-46616-9_10.

[117] Rui Wui, Conner Scully-Allison, Chase Carthen, Andie Garcia, Christopher Lewis, Ronn Siedrik Quijada, Jessica Smith, Sergiu M. Dascalu, and Frederick C. Harris, Jr. vFirelib: a GPU-based fire simulation library and fire data visualization. *Submitted*, 2019.

[118] Hong Xue, Juay Choy Ho, and Y.M Cheng. Comparison of different combustion models in enclosure fire simulation. *Fire Safety Journal*, 36(1):37–54, 2001. ISSN: 0379-7112. DOI: https://doi.org/10.1016/S0379-7112(00)00043-6. URL: https://www.sciencedirect.com/science/article/pii/S0379711200000436.

[119] Powen Yao, Vangelis Lympouridis, Tian Zhu, Michael Zyda, and Ruoxi Jia. Punch typing: alternative method for text entry in virtual reality. In *Symposium on Spatial User Interaction, Virtual Event, Canada*, SUI '20, New York, NY, USA. Association for Computing Machinery, 2020. ISBN: 9781450379434. DOI: 10.1145/3385959.3421722. URL: https://doi.org/10.1145/3385959.3421722.

[120] Ali Shahidi Zandi, Manouchehr Javidan, Guy A. Dumont, and Reza Tafreshi. Automated real-time epileptic seizure detection in scalp eeg recordings using an algorithm based on wavelet packet transform. *IEEE Transactions on Biomedical Engineering*, 57(7):1639–1651, 2010. DOI: 10.1109/TBME.2010.2046417.

Appendix A

UNR IRB Application

University of Nevada, Reno
Institutional Review Board
Part I, Cover Sheet

Last edited by: Christopher Lewis

Last edited on: March 15, 2022

[click for checklist]

[1888910-1] A comparison of typing methods in virtual reality

Answer all questions on this form completely, include attachments and obtain signatures of Principal Investigator and the Responsible official prior to final submission on IRBNet.

I. Principal Investigator

Name: Frederick Harris, PhD

Institution:

Department: College of Engineering

Telephone: (775) 784-6571 **Email:** fred.harris@cse.unr.edu

Address: 1664 N Virginia St, WPEB-437, Reno, NV 89557-0171, USA

COI Disclosure: Any financial interests related to this study?

- ☐ Yes
- ☑ No

If yes, **COI Disclosure Explanation:**

Could any external entity benefit financially from the results of this study?

- ☐ Yes
- ☑ No

II. Co-Investigator(s) or Research Team Member(s) N/A ☑

Name:

Department:

Institution:

Telephone: **Email:**

COI Disclosure: Any financial interests related to this study?

- ☐ Yes
- ☐ No

If yes, **COI Disclosure Explanation:**

III. Student Investigator(s) N/A ☐

Name: Christopher Lewis

Institution:

Telephone: 7758308660 **Email:** christopher_le1@nevada.unr.edu

COI Disclosure: Any financial interests related to this study?

- ☐ Yes
- ☑ No

If yes, **COI Disclosure Explanation:**

IV. Project Information

Research Type:

- ☐ Biomedical
- ☑ Social Behavioral/Educational

Photographing or Video Recording:

- ☐ Yes
  Upload the Photo Release form.
- ☑ No

Identifiable Information from Education Records:
Note that accessing education records for research purposes invokes FERPA regulations. In the protocol, address how records are accessed, whether researchers will access directory information only, and how written permission will be collected from students or parents for minor students. For more information on FERPA, see IRB Policy 76.

- ☐ Yes
- ☑ No

Research Location:

- ☐ VA Sierra Nevada Healthcare System
- ☐ Saint Mary's Regional Medical Center
- ☐ Renown Health
- ☑ UNR Campus
- ☐ Other -

International Research:

- ☐ Yes
  For international research, reference IRB Policy 575 for details to address in the protocol.
- ☑ No

If yes, **specify the countries:**

Renown Health Research Locations:

- ☐ Renown Regional
- ☐ Renown Pharmacy

- ☐ Renown South Meadows
- ☐ Renown Pregnancy Center
- ☐ Renown Outpatient Clinic
- ☐ Renown Urgent Care
- ☐ Renown Imaging
- ☐ Renown Lab
- ☐ Renown Emergency Room
- ☐ Renown Skilled Nursing
- ☐ Renown Hospice Care
- ☐ Renown Home Health
- ☐ Renown Rehabilitation

Requested Review Path:

- ☐ Expedited IRB Review
  Complete Protocol - Social Behavioral Educational Research and Records Research or Protocol - Biomedical Research
- ☐ Full Board Review
  Complete Protocol - Social Behavioral Educational Research and Records Research or Protocol - Biomedical Research
- ☑ Exempt Review
  Complete Protocol - Social Behavioral Educational Research and Records Research
- ☐ Requesting a determination about whether a project is human research
  Complete Request for Human Research Determination
- ☐ Requesting authorization to use an external IRB
  Complete Request to Use an External IRB
- ☐ Reporting emergency use of an FDA-regulated drug or device
  Complete Emergency Use Investigational Drug or Device
- ☐ Review of a Humanitarian Use Device
  Complete Protocol - Humanitarian Use Device for Treatment or Diagnosis
- ☐ Research involving existing records or specimens
  Complete Protocol - Social Behavioral Educational Research and Records Research

Risk Level:

- ☑ Minimal risk
- ☐ Greater than minimal risk (requires full board review)
- ☐ No known risk

Involvement of Vulnerable Populations:

- ☑ N/A, research will not involve vulnerable populations
- ☐ Pregnant women and fetuses
  For research with pregnant women and fetuses, reference IRB Policy 210 and 211 for details to address in the protocol.
- ☐ Prisoners
  Complete Research with Prisoners
- ☐ Children (persons under 18 years of age)
  For research with children, reference IRB Policy 230 for details to address in the protocol.
- ☐ Adults with impaired decision-making capacity
  For research with adults with impaired decision-making capacity, reference IRB Policy 240 for details to address in the protocol.
- ☐ People who do not speak English

For research with people who do not speak English, reference IRB Policy 250 for details to address in the protocol.

V. Funding Information N/A ☑

Sponsor Type:

- ☐ Federal Government
- ☐ Industry Sponsor
- ☐ Departmental
- ☐ Other:
- ☐ Other Government (State/Local)
- ☐ Other Private Funds
- ☐ Subcontract

Sponsor Name:

Grant/Contract Title and Number:

VI. Federal Agencies with Additional Requirements to Protect Human Participants

Please see the "Instructions to Researchers" section at the end of this form for a list of required supplemental forms and relevant policies.

- ☐ DoD
- ☐ DoE
- ☐ DoEd
- ☐ DoJ or NIJ
- ☐ EPA
- ☐ NSF
- ☐ VA
- ☑ N/A

VII. FDA-Regulated Research

- ☑ N/A, research does not involve drugs or devices
- ☐ Drug research

Trade Name	Generic Name

- ☐ Device research

Name of Device	Device Manufacturer

VIII. External Committee Approvals

- ☐ Thesis or Dissertation Committee
- ☐ Radiation Safety Committee

- ☐ Biosafety Committee
- ☐ Other:
- ☑ N/A

INSTRUCTIONS TO RESEARCHERS

[top]

You have completed Part I of the application process. **Preview** Part I and correct if needed. Print the last page so you have the list of the researcher forms and additional regulatory requirements expected for this research. Click **Save and Exit**. **Add** the remaining required documents (listed below or referenced in the researcher forms/applications), **address** the necessary regulatory and policy requirements in the protocol and other project documents, and then the PI should electronically **Sign** and **Submit** the project. Make sure to upload training documentation for all researchers listed on this form.
If you have any questions, refer to the IRBNet pages of the Research Integrity website.

Additional required researcher forms and policies/regulations:

- Complete Protocol - Social Behavioral Educational Research and Records Research

Appendix B

UNR IRB Application

University of Nevada, Reno
Research & Innovation

Research Integrity
Tel: (775) 327-2368
Web: unr.edu/research-integrity

Protocol – Social Behavioral Educational Research and Record Research
1888910-2, Dr. Frederick C. Harris Jr.

Background:
This research needs to be done due to the fact that a good method for text input in virtual reality (VR) does not currently exist. A few other research articles exist on various VR input methods. The most standard form of VR input is with a point and click type system, this can be seen in something like Google Earth **[1]**. This is where the user aims their VR controller at a virtual 2D keyboard and selects a key one at a time. Another input method is "punch typing" **[2]**. This is where a 3D keyboard, generally in the form of spheres, exists in front of the user. The user would take their controller or VR glove and virtually hit or collide with the spheres. Another form of input methods is the "two-thumb touchpad" **[3]**. This input method uses the trackpad on both VR controllers. Each controller has a corresponding cursor on a 2D keyboard. The user would navigate, using the trackpad to move the cursor, to the key they want to press and select it. Finally, the last input method I'll be talking about is the "drum-like keyboard" **[4]**. This input method consists of a 3D virtual keyboard in front of the user and virtual drumsticks extending off from each of the user's VR controllers. To input text, the user makes the virtual drumsticks collide with any key on the 3D keyboard. All of these input methods generally have associated WPM and errors per minute (EPM) in the research where they are created, generally found through user studies. A few also have had comparison studies **[5]** ran on them to determine accuracy and effectiveness for each input method. All of this research shows that the average WPM for each input method never passed 25 WPM. This is a huge downgrade from the average WPM found on modern keyboards, which is generally to be assessed at 40 WPM **[6]**. The fastest input method found was the "drum-like keyboard" at 25 WPM average. There has also been some research done on the effectiveness of physical keyboards to be used inside of VR **[7]**, and while effective, the solution is bulky and removes some of the unique characteristics of VR. Thus, a fast input method built inside of VR is required to further this line of research.

The two input methods that this research will be comparing in the user study are as follows: a modified dictation/speech-to-text algorithm; and a simplified modified dictation algorithm with a companion modified "drum-like keyboard". Dictation is really quick as far as input methods go. Due to the fact that the average talking speed is 168 WPM **[8]**, I believe that a dictation algorithm will have a very easy time passing the previous average WPM found in VR studies and other input methods. I also believe that due to the hands-free nature of dictation algorithms, this input method will help those with disabilities that affect their fine motor control, like cerebral palsy, physical Tourette's syndrome, and Parkinson's disease, since most other input methods require a degree of fine motor control.

The issue with dictation algorithms; however, lies in the fact that they generally only output words and whole sentences. This limitation makes it very hard for users to type out things like URLs, Emails, and abbreviations, which is the advantage of having a full keyboard. This is why I've titled this input method as a "modified dictation algorithm", the modification comes into play in typing things like special characters or single letters. The modified dictation algorithm allows the user to select a custom command phrase (for now I have it set to a default of "command"). This command phrase allows the user to switch what type of input they want. If the user says something along the lines of, "command dot" the output text will be just ".", while if they said the same thing without the command phrase, the output text will be "dot".

The simplified modified dictation algorithm does not have the special characters as a part of the command list, but it does have the ability to write out single letters, just like the modified dictation algorithm. To do something like that you would say, "command i" which will have the output text of "i", if you removed the command phrase the output text would be "eye". "command capital" will switch between uppercase and lowercase letters.

The “modified drum-like keyboard” works exactly like the regular “drum-like keyboard”, except it’s primary focus is on displaying special characters, which pairs well with the “simplified modified dictation algorithm”.

Due to all of these novel input methods, the research is required to fully grasp what use cases each input method fits into and just how fast people can input text into VR. This is becoming increasingly more important as VR is starting to be used in training simulations in corporate environments, as well as commercial use.

Study Aims/Objectives:

1. Create a faster way to input text in VR
2. Calculate words-per-minute, characters-per-minute, and errors-per-minute of two novel input methods
3. Compare input methods to determine validity, usability, and effectiveness of both

Study Population:
The participants must have an age greater than 18. There are no particular ethnic/gender inclusions or exclusions. Inclusion criteria are that the participant must be able to speak, read, and write English at a working level. The participant must also have full working use of their hands and arms, and be able to grip and hold a VR controller.

Vulnerable Populations:
N/A

Sample Size:
Sample size should be 20. This number was reached by asking a stat’s professor at UNR. The professor informed us that it should be about 10 people per input method we wanted to compare.

Recruitment Process:
The main way to identify potential participants will be to have email recruitment. The participants will be able to schedule their turn at the study individually. The recruitment email is uploaded as “Recruitment_Email.docx” in IRBNet.

Screening Procedures:
The inclusion criteria for needing to be able to speak, read, and write English at a working level will be satisfied by receiving an email that confirms the time and location of their study.

The inclusion criteria for needing to have working use of their hands and arms, and be able to hold/grip a VR controller will be assessed before the main scenario of the user study begins in a testing scenario that mimics the main scenario. This testing scenario exists to allow the participant to get acclimated to VR and it will allow the researchers to answer questions about the input methods, without taking up time in the main scenario (and thus lowering their WPM and perhaps increasing their EPM without cause). The user will not be timed in the testing scenario and they can leave it at any point. If the user can not physically hold the VR Controllers or use them accurately enough to type on the modified drum-like keyboard, they will not be included in the user study.

Informed Consent Process:
Before the participant puts on the VR headset and starts the user study, they will be presented with a printed version of the file titled, “Consent_Form.doc”. The researcher will then verbally read out the “Introduction” section of this document. The researcher will then ask the participant to read through the rest of the document, and then sign the document if they understand it’s contents. The researcher will then move away from the user to give them privacy and to set up the VR environment for a successful run through of the application.

When/if the user has signed the consent form, it will be collected into a locked file cabinet in WPEB436, which is in itself a locked laboratory. This room only allows approved individuals to enter via verification of their wolf card. Approved individuals are determined by the PI, Dr. Frederick C. Harris Jr.

Data Collection Procedures:

1. The participant will be asked to complete the consent form (see the informed consent process section in this document for detailed information on this process)
2. The participant will be asked to complete the pre-test survey titled “Pre_Test_Survey.doc”. The researcher will also move away from the participant as described in the informed consent process section.
3. The participant will then be helped into the VR headset (head-mounted display, HMD) by the researcher
4. The researcher will then hold up the VR controllers in front of the participant
5. The researcher will ask if the participant can see the VR controllers in the virtual world
6. If the participant can see the controllers, the researcher will ask the participant to grab them
7. If the participant can’t see the controllers, the researcher will assess the condition of the controllers to correct any mistakes with setup, once accomplished go to step 4
8. The researcher will turn on the application
9. The participant will then select the, “Test scenario” from the menu that displays onto their screen.
10. The researcher will then ask the user to complete various sample texts with the drum-like keyboard and with the dictation feature.
11. The researcher will ask/answer the participant if they have any/more questions before they proceed to the timed user study.
12. The researcher will then tell the participant to hit a button, causing he user to go back into the same menu as step 9
13. The participant will then go into the “Main Scenario”
14. The researcher instructs the participant to input the text that they see displayed in VR.
15. The application then monitors the participant’s words per minute (WPM), characters per minute (CPM), and errors per minute (EPM)
16. The application also monitors when the participant is using dictation to input text, when the participant is using the drum-like keyboard’s special characters, and when the participant is using the drum-like keyboard’s QWERTY keyboard to input text. This input is tied to which input type the user is typing: a URL, email, sentence, or paragraph.
17. When the participant has finished with one input type, they will be automatically swapped to another sample text of the same input type, but they will also be switched to the other input method.
18. When the participant has finished with one input type and with both input method, another input type is then chosen to be completed
19. Steps 17 and 18 repeats until all of the input types are completed with both input methods
20. Once done with all input types and input methods, the participant will then be helped out of the HMD
21. Once out of the HMD, the researcher will give them the System Usability Scale (SUS) for Dictation, “SUS.doc”
22. The researcher will give them the same privacy as in steps 1 and 2.
23. Once the participant has finished the SUS for Dictation, they will be given the SUS for Dictation + Drum-like keyboard (The same document, “SUS.doc”)
24. The researcher will give them the same privacy as in steps 1 and 2.
25. Once finished, the researcher will give them the “Post_Test_Survey.doc” and the “VR Simulator Sickness Questionnaire SSQ.pdf” to fill out.
26. The researcher will give them the same privacy as in steps 1 and 2.
27. Once finished, the participant will be asked to leave the room and the user study for that participant will be over.

Study Duration/ Study Timeline:
Each participant will contact the researcher approximately three times. First, to ask the researcher to join the study (email). The researcher will then respond with availability (email). Second, to confirm availability for a specific hour of time (email). The researcher will then respond with confirmation that the participant can come at that time as well as with location information (email). Third, to actually complete the user study. This will be the only time the researcher and the participant should meet physically for the user study. Again, the duration of this meeting should only take about an hour. Approximate end date of the study is June 30th, 2022.

Study Locations:
The location in which the research will take place is in The University of Nevada, Reno's William Pennington Engineering Building (WPEB), room number 436. This is the PI's laboratory. This location is behind a locked door with a "Wolf Card" scanner for access. The door will be open when the researcher expects the participant to make entry easier and obvious for the participant. The study will take place at a desk in that room.

International Research:
N/A

Participant Compensation:
N/A

Risk to Participants:
There is a slight risk of discomfort caused by motion sickness in VR. This is a standard risk for VR, thus it is hard to minimize; however, the participants will be asked to remain sitting during their user study, to facilitate less chances of motion sickness. This risk is generally caused by the inner ear due to a disconnection between what you are seeing your body do (virtually) and what you are actually doing, thus we are also trying to match the virtual movements to the physical ones as best we can.

There is also a very slight risk of discomfort caused by wearing the HMD. This risk is minimized by encouraging the participant to tell the researcher if they'd like to pause or stop the study. This risk is also minimized by keeping the VR portion of the study under an hour.

Benefits to Participants:
This research does not present any direct benefit to the participants. However, the research provides an opportunity to gain a better understanding of VR applications and typing in VR

Privacy of Participants:
The participant's privacy will be protected during recruitment by not telling anyone who is going to be in the study. This means only one person at a time will be emailed by the researchers and any initial recruitment will be done via generalized emails to multiple people at a time. Informed consent will be handled on an individual basis right before the start of the study. The informed consent with their signature will then be placed in a locked cabinet in a locked room. The data being collected via the VR application (WPM, CPM, EPM, etc.) will be stored in a file system protected by the researcher's individual, private, username and password.

The settings in which the participant will be interacting with a researcher are in two locations. The first is via Email. The second is in person in a largely quiet room that may or may not have additional researchers (that are not involved with the study). The room will have an open door for the duration of the study. The researcher may have to touch the participant's head in order to help them get into the VR headset, but only doing so after asking if the participant needs help in putting it on. Anytime the participant needs to sign, fill out, or read a document the researcher will move away as to show the participant that they are not observing their answers and giving them privacy to read and take as much time as they would like.

The only time the participant's name will be recorded is in the informed consent paper. This paper will not be linked in any way to the participant's application data, nor their pre/post survey answers, thus ensuring complete anonymity.

Data Management and Confidentiality:
The informed consent paperwork will be kept in a locked cabinet in a locked location until all of them have been collected that are necessary for the study. Once that happens, the PI will take the paperwork and seal it in a box, to be shredded at a later date per UNR policy. The data collected in the pre/post study, and in the application itself will be collected, analyzed for significant scientific findings, and then compressed and encrypted. This encrypted data will then be given to the PI for this project to store until destroyed, per UNR policy.

The only people who will have access to the data will be the researcher, Christopher Lewis, and the PI, Dr. Frederick C. Harris Jr.

None of the data will be linked to individuals, except the informed consent form, which will not be linked to any of the data either.

Approach to Analysis:
We plan to analyze the data for average WPM, CPM, and EPM. We also plan to analyze the approaches of each individual for input method selection versus input type, to hopefully create a trend between what input method participants enjoy using for each input type. We also plan to cross reference the post survey data to see if they participants were aware that they wanted to use one input method over another. The SUS questionnaire will be analyzed for standard deviation and overall usability of each input method.

References:
[1] *Google Earth VR*. https://arvr.google.com/earth/. Accessed 17 Apr. 2022.

[2] Yao, Powen, et al. "Punch Typing: Alternative Method for Text Entry in Virtual Reality." *Symposium on Spatial User Interaction*, ACM, 2020, pp. 1–2. *DOI.org (Crossref)*, https://doi.org/10.1145/3385959.3421722.

[3] Son, Jeongmin, et al. "Improving Two-Thumb Touchpad Typing in Virtual Reality." *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, ACM, 2019, pp. 1–6. *DOI.org (Crossref)*, https://doi.org/10.1145/3290607.3312926

[4] Boletsis, Costas, and Stian Kongsvik. "Text Input in Virtual Reality: A Preliminary Evaluation of the Drum-Like VR Keyboard." *Technologies*, vol. 7, no. 2, Apr. 2019, p. 31. *DOI.org (Crossref)*, https://doi.org/10.3390/technologies7020031

[5] Boletsis, Costas, and Stian Kongsvik. "Controller-Based Text-Input Techniques for Virtual Reality: An Empirical Comparison." *International Journal of Virtual Reality*, vol. 19, no. 3, Oct. 2019. *DOI.org (Crossref)*, https://doi.org/10.20870/IJVR.2019.19.3.2917

[6] *Average Typing Speed Infographic — Ratatype*. https://www.ratatype.com/learn/average-typing-speed/. Accessed 17 Apr. 2022

[7] Pham, Duc-Minh, and Wolfgang Stuerzlinger. "HawKEY: Efficient and Versatile Text Entry for Virtual Reality." *25th ACM Symposium on Virtual Reality Software and Technology*, ACM, 2019, pp. 1–11. *DOI.org (Crossref)*, https://doi.org/10.1145/3359996.3364265

[8] "Do You Speak Toofast or Toooo S-L-O-W-L-Y?" *Successfully Speaking*, https://successfully-speaking.com/blog/2016/6/27/do-you-speak-toofast-or-toooo-s-l-o-w-l-y. Accessed 17 Apr. 2022

Appendix C

UNR IRB Consent Form

University of Nevada, Reno
Social Behavioral or Educational Research Consent Form

Title of Study:	**A comparison of typing methods in virtual reality**
Principal Investigator:	**Frederick C. Harris Jr., PhD**
Co-Investigators /	**Christopher J. Lewis**
Study Contact:	**Christopher J. Lewis**
Study ID Number:	**1888910-1**
Sponsor:	**N/A**

SUMMARY OF KEY ELEMENTS:

Introduction

You are being invited to participate in a research study. Before you agree to be in the study, read this form carefully. It explains why we are doing the study; and the procedures, risks, discomforts, benefits and precautions involved.

At any time, you may ask one of the researchers to explain anything about the study that you do not understand.

You do not have to be in this study. Your participation is voluntary. If you do not agree to participate, you will receive the care/education you would have received if the study was not taking place.

Take as much time as you need to decide. If you agree now but change your mind, you may quit the study at any time. Just let one of the researchers know you do not want to continue.

Why are we doing this study?

We are doing this study to determine the overall performance and usability of a novel approach to typing in virtual reality, while also examining possible improvements that can be made.

Why are we asking you to be in this study?

We are asking you to be in this study because you are an adult who can read and write English and also has the ability to move their body in virtual reality.

How many people will be in this study?

We expect to enroll around 20 participants at the University of Nevada, Reno.

What will you be asked to do if you agree to be in the study?

If you agree to be in this study you will be asked to complete a pre-test survey, post-test survey, and a simulator sickness questionnaire at the beginning and end of the experiment along with a series of tasks which help to assess overall performance and usability of the application.

The pre-test survey will ask you questions regarding your age, gender, highest completed level of education, and various questions relating to your familiarity with key components of the

application. The simulator sickness questionnaire will ask you if you are experiencing any symptoms associated with simulator sickness and how severely they are affecting you.

The tasks of the study will be performed while wearing a VR headset and holding VR controllers. These tasks include a period of teaching you how to use the controllers to type, and how to speak commands into the VR headset for voice recognition, as well as, a period of you typing desired text that will show up on screen as fast as you can.

The post-test survey will ask questions regarding the participant's thoughts on usability and design of the application along with any improvements that can be made.

How long will you be in the study?

The study will take about an hour of your time.

What are your choices if you do not volunteer to be in this research study?

If you decide not to be in the study, tell the investigator and you will be allowed to leave.

What if you agree to be in the study now, but change your mind later?

You do not have to stay in the study. You may withdraw from the study at any time by leaving the room after informing the investigator.

What if the study changes while you are in it?

If anything about the study changes or if we want to use your information in a different way, we will tell you and ask if you want to stay in the study. We will also tell you about any important new information that may affect your willingness to stay in the study.

Is there any way being in this study could be bad for you?

Some users of virtual reality experience discomfort, nausea, headaches, and eye strain. If at any point in time you begin to experience any of these symptoms, please remove the virtual reality headset. We can continue testing or stop altogether.

What happens if you become injured because of your participation in the study?

In the event that this research activity results in an injury, treatment will be available. This includes first aid, emergency treatment, and follow-up care as needed. Care for such injuries will be billed in the ordinary manner to you or your health insurance carrier.

Will being in this study help you in any way?

Probably not, except it will add to your experience with using VR applications.

Who will pay for the costs of your participation in this research study?

No costs are associated with participation in this study.

Will you be paid for being in this study?

You will not receive any payment for being this study.

Who will know that you are in in this study and who will have access to the information we collect about you?

The researchers, the University of Nevada, and the Reno Institutional Review Board will have access to your study records.

How will we protect your private information and the information we collect about you?

We will treat your identity with professional standards of confidentiality and protect your private information to the extent allowed by law.

We will not use your name or other information that could identify you in any reports or publications that result from this study.

Do the researchers have monetary interests tied to this study?

The researchers and/or their families have no monetary interests tied to this study.

Whom can you contact if you have questions about the study or want to report an injury?

At any time, if you have questions about this study or wish to report an injury that may be related to your participation in this study, contact Frederick Harris at (775) 784-6571 and/or Christopher Lewis at (775) 830-8660.

Whom can you contact if you want to discuss a problem or complaint about the research or ask about your rights as a research participant?

You may discuss a problem or complaint or ask about your rights as a research participant by calling the University of Nevada, Reno Research Integrity Office at (775) 327-2368. You may also use the online *Contact the Research Integrity Office* form available from the *Contact Us* page of the University's Research Integrity Office website.

Agreement to be in study

If you agree to participate in this study, you must sign this consent form. We will give you a copy of the form to keep.

Participant's Name Printed	
Signature of Participant	Date
Signature of Person Obtaining Consent	Date

Appendix D

UNR IRB Pre-Test Survey

University of Nevada, Reno
Social Behavioral Research

Title of Study:	**A comparison of typing methods in virtual reality**
Principal Investigator:	**Frederick C. Harris Jr., PhD**
Co-Investigators /	**Christopher J. Lewis**
Study Contact:	**Christopher J. Lewis**
Study ID Number:	**1888910-1**
Sponsor:	**N/A**

Pre Test Survey

Participant ID #:_____

Please Enter Your Age:_____

What is your gender? _____

What is your highest level of completed education? _____

Please rate your familiarity with Virtual Reality (VR).
(1 – Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Please rate your familiarity with Motion Tracking Hardware and Software.
(1 – Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Please rate your familiarity with Video Games or Similar Forms of Electronic Entertainment.
(1 – Not Familiar, 5 - Very Familiar)

Not Familiar	1	2	3	4	5	Very Familiar

Appendix E

UNR IRB System Usability Scale

System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Appendix F

UNR IRB Post-Test Survey

University of Nevada, Reno
Social Behavioral Research

Title of Study:	**A comparison of typing methods in virtual reality**
Principal Investigator:	**Frederick C. Harris Jr., PhD**
Co-Investigators /	**Christopher J. Lewis**
Study Contact:	**Christopher J. Lewis**
Study ID Number:	**1888910-1**
Sponsor:	**N/A**

Post Test Survey

Please rate how comfortable you were during the test.
(1 – Very Uncomfortable, 5 – Very Comfortable)

Very Uncomfortable	1	2	3	4	5	Very Comfortable

Please rate how easy it was to input each text.
(1 – Very Difficult, 5 - Very Easy)

Very Difficult	1	2	3	4	5	Very Easy

Please rate the intuitiveness of the user interface.
(1 – Not Intuitive, 5 - Very Intuitive)

Not Intuitive	1	2	3	4	5	Very Intuitive

Please rate the usefulness of the typing methods in general.
(1 – Not Useful, 5 - Very Useful)

Not Useful	1	2	3	4	5	Very Useful

Please rate your overall user experience.
(1 – Very Bad, 5 - Very Good)

Very Bad	1	2	3	4	5	Very Good

What potential improvements, if any, would make the application more useful or easy to use?

_____
_____
_____
_____
_____
_____
_____
_____

Based on your experience, what are some other virtual reality typing methods you would find useful?

_____
_____
_____
_____
_____
_____
_____
_____

Based on your experience, do you have a preference for any typing method?

_____
_____
_____
_____
_____
_____
_____
_____

Based on your experience, do you think any typing method was better for one input type or another (email, url, sentence, or paragraph)?

_____
_____
_____
_____
_____
_____
_____
_____

Please write any other comments or observations that you might have.

_____
_____
_____
_____
_____
_____
_____
_____

Appendix G

UNR IRB Simulator Sickness Questionnaire

No_____ Date_____

SIMULATOR SICKNESS QUESTIONNAIRE

Kennedy, Lane, Berbaum, & Lilienthal (1993)***

Instructions : Circle how much each symptom below is affecting you right now.

1. General discomfort	None	Slight	Moderate	Severe
2. Fatigue	None	Slight	Moderate	Severe
3. Headache	None	Slight	Moderate	Severe
4. Eye strain	None	Slight	Moderate	Severe
5. Difficulty focusing	None	Slight	Moderate	Severe
6. Salivation increasing	None	Slight	Moderate	Severe
7. Sweating	None	Slight	Moderate	Severe
8. Nausea	None	Slight	Moderate	Severe
9. Difficulty concentrating	None	Slight	Moderate	Severe
10. « Fullness of the Head »	None	Slight	Moderate	Severe
11. Blurred vision	None	Slight	Moderate	Severe
12. Dizziness with eyes open	None	Slight	Moderate	Severe
13. Dizziness with eyes closed	None	Slight	Moderate	Severe
14. *Vertigo	None	Slight	Moderate	Severe
15. **Stomach awareness	None	Slight	Moderate	Severe
16. Burping	None	Slight	Moderate	Severe

* Vertigo is experienced as loss of orientation with respect to vertical upright.

** Stomach awareness is usually used to indicate a feeling of discomfort which is just short of nausea.

Last version : March 2013

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

Simulator Sickness Questionnaire***

Kennedy, Lane, Berbaum, & Lilienthal (1993)***

Validation of the French-Canadian version of the SSQ developed by the UQO Cyberpsychology Lab :

- Total : items 1 to 16 (scale of 0 to 3).
  - « *Nausea* » : items 1 + 6 + 7 + 8 + 12 + 13 + 14 + 15 + 16.
  - « *Oculo-motor* »: items 2 + 3 + 4 + 5 + 9 + 10 + 11.

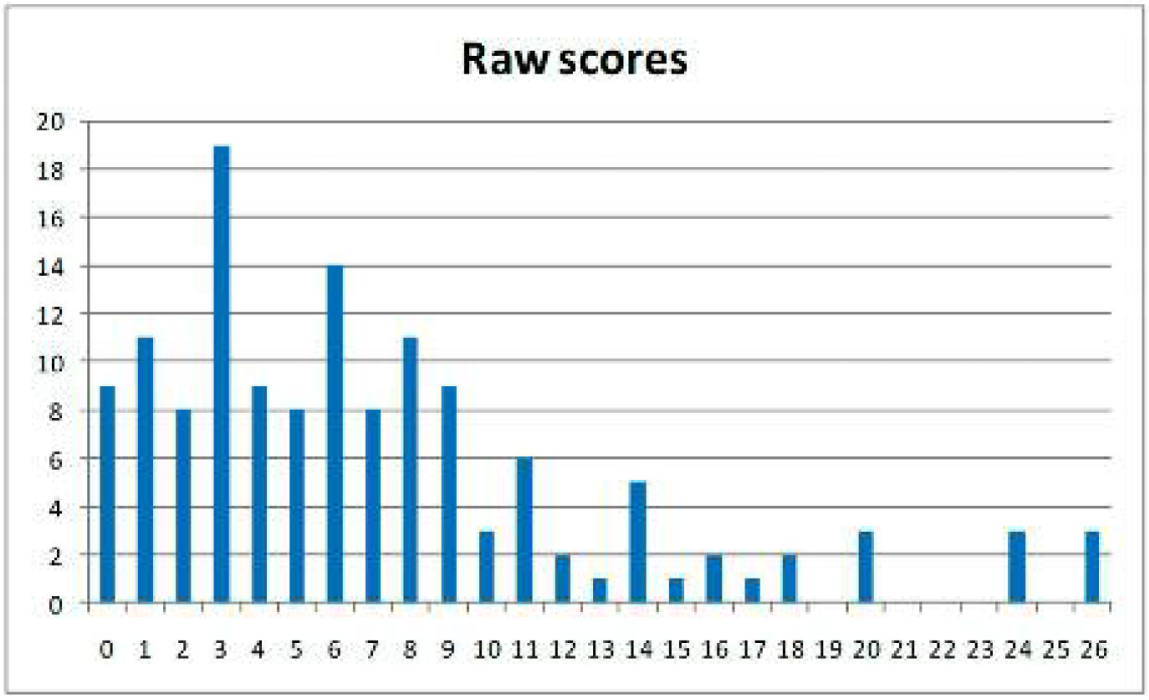
Please refer to the following articles for more information about the French-Canadian validated version :

BOUCHARD, S., Robillard, & Renaud, P. (2007). Revising the factor structure of the Simulator Sickness Questionnaire. Acte de colloque du *Annual Review of CyberTherapy and Telemedicine*, *5*, 117-122.

BOUCHARD, S., St-Jacques, J., Renaud, P., & Wiederhold, B.K. (2009). Side effects of immersions in virtual reality for people suffering from anxiety disorders. *Journal of Cybertherapy and Rehabilitation*, 2(2), 127-137.

BOUCHARD, S. Robillard, G., Renaud, P., & Bernier, F. (2011). Exploring new dimensions in the assessment of virtual reality induced side-effects. *Journal of Computer and Information Technology*, *1*(3), 20-32.

Based on results from Bouchard, St-Jacques, Renaud, & Wiederhold (2009), below are the mean scores reported:



Note. For the original scoring version, consult : Kennedy, R.S., Lane, N.E., Berbaum, K.S., & Lilienthal, M.G. (1993). Simulator Sickness Questionnaire: An enhanced method for quantifying simulator sickness. *International Journal of Aviation Psychology*, *3*(3), 203-220.

Appendix H

UNR IRB Participant Recruitment Email

Recruitment Email Script

General Recruitment Email:

Good [morning, afternoon, etc.],

I am conducting a research study to assess the performance and usability of two virtual reality (VR) typing methods. I am currently looking for 20 participants to conduct a user study to advance this research. Participants in the study will be asked to perform a number of tasks while wearing both a VR headset and holding VR controllers. These tasks include getting comfortable with typing in VR using both a virtual keyboard and speech-to-text, and then taking a typing test by trying to type a URL, email, sentence, and paragraph as fast as you can. The entire process should take approximately an hour to complete. The user study will be performed in WPEB Lab Room 436. If you are interested in participating in this user study, please email Christopher Lewis at christopher_le1@nevada.unr.edu to set up a time that works best for you to complete the study.

Sincerely,

Christopher Lewis

Personal Recruitment Email:

Dear [participant],

I am conducting a research study to assess the performance and usability of two virtual reality (VR) typing methods. I am currently looking for 20 participants to conduct a user study to advance this research. Participants in the study will be asked to perform a number of tasks while wearing both a VR headset and holding VR controllers. These tasks include getting comfortable with typing in VR using both a virtual keyboard and speech-to-text, and then taking a typing test by trying to type a URL, email, sentence, and paragraph as fast as you can. The entire process should take approximately an hour to complete. The user study will be performed in WPEB Lab Room 436. If you are interested in participating in this user study, please email Christopher Lewis at christopher_le1@nevada.unr.edu to set up a time that works best for you to complete the study.

Sincerely,

Christopher Lewis