THE DESIGN OF VIRTUAL REALITY BASED DATA VISUALIZATION AND USER INTERFACE DESIGN IN A SEMI-AUTOMATED CYBER-SECURITY RESEARCH APPLICATION

A Thesis
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Santipab Tipparach

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

December 2019

Fargo, North Dakota

North Dakota State University
Graduate School

Title

THE DESIGN OF VIRTUAL REALITY BASED DATA VISUALIZATION AND USER INTERFACE DESIGN IN A SEMI-AUTOMATED CYBER-SECURITY RESEARCH APPLICATION

By

Santipab Tipparach

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Jeremy Straub
Chair

Dr. Simone Ludwig

Dr. Mark Nawrot

Approved:

December 16, 2019
Date

Dr. Kendall Nygard
Department Chair

ABSTRACT

Virtual Reality is currently an affordable and consumer ready technology used by many in the games and interactive media industry, however unlike the user interface standards in mobile, PCs, and Macs, VR UI design can vary in complexity and usability. VR has many times been linked in films, TV shows, and animation as a method for navigating through cyberspace. It has been portrayed to be involved in the process of hacking a computer on some network. This study will look at approaches to developing a UI system using cyber-security research applications as a basis for designing a framework. Throughout, this research will analyze the different approaches to UI design and data visualization, extract relevant information, and find out what approaches will help improve the VR software front end design.

ACKNOWLEDGEMENTS

I would like to thank Wren Erickson who helped me test, collect data, and come up with ideas to simulate the cyber world, Wyly Andrews for helping with the search number feature for IP addresses, Isaac Burton for providing helped advise the details of simulating cyber-attacks, and Dr. Jeremy Straub for helping to come up with the idea for the project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDIX FIGURES

1. INTRODUCTION

The most interesting application of Virtual Reality technology is not in how closely it can mimic real life interactions, but how it can create new experiences. VR is still not a very widely used technology, despite being more than 3 years since the Oculus Rift and HTC Vive were first publicly released as the major competitors in the VR industry. A lot of the problems that come with VR are mostly due to its bulky hardware, requiring a powerful PC. Unlike televisions or mobile phones, the presentation of VR is a delicate balance of usability and comfort. Although VR is a significant leap in display and interactive technology, it has not yet replaced existing flat screens displays, but it provides a glimpse into a future where the user can live inside a virtual environment.

Often found in media is the depiction of cyber-attacks or hacking a system via vibrant visuals or a dramatic scene of the hackers racing against time to accomplish their mission. One of the biggest influences on this study, as seen in Figure 1, comes from a scene in Hackers where a hacker uses virtual reality to navigate through a database deleting files [28].

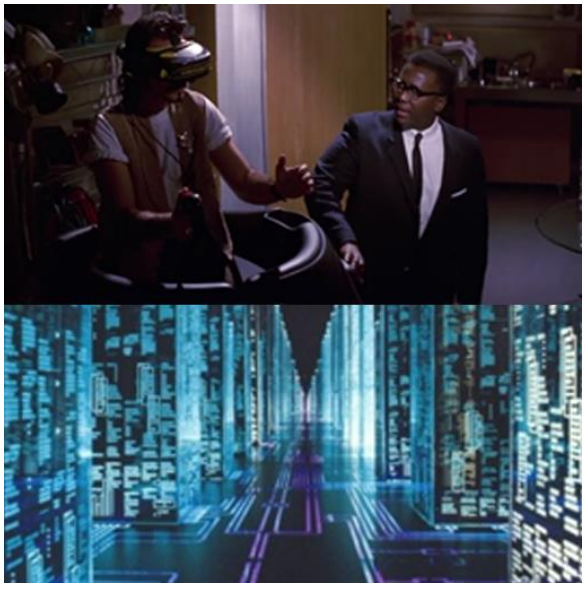


Figure 1. Scene from Hackers that shows cinematic representation of how hacking can be carried out in VR [28].

In cinema, hacking visualization is often used to dramatize something that is purely digital, a concept that is hard for the audience to visualize. These scenes often involve some representation of cyberspace. While in films and other media, cinematic hacking helps the viewer visualize the event taking place, this research aims at representing an interactive cyberworld to help the user understand the cyberspace they are attempting to exploit.

The study's primary goal is to investigate various methods for developing UI in VR and represent information within the virtual world. The development of this framework was centered around the theme of cyber-security research: a concept VR application that simulates the process of automated cyber-attacks. The concept of this application is to create a framework that would host a few different components that would help benchmark a VR application for cyber-attacks. These components include: visual network graphs, interacting with geographic data, menu navigation, object selection, and event executions. Each of these components will be touched upon, however the focus of the research is the core UI system and the different use cases that various UI models can be used effectively.

Initially, the research took a deep dive into the process of developing a cyber-security research application. The first goal was to investigate the process for making an application in VR viable for non-cyber-security experts to be able to easily use to visualize and understand the steps taken behind a cyber-attack. The top-down view of this concept was to develop an application that could automate an entry-level automated network analysis and host exploitation tool. This tool would be used to study network vulnerability using existing tools that were traditionally ran in separate various applications or within the Linux terminal. The development investigated various methods for visualizing the data in a graph form. Then it took the approach of simulating an exploit on virtual host machines. The process for this led to a need for a clearer

understanding of how the user would interact with the system. Thus, the study shifted its focus towards the development of the UI.

In the development of cyber-security tools in VR, one aspect stood out as being abstract and difficult to justify. That aspect was how the user would interact with the data. Solving the problem of user-data interaction in a VR environment would require a comparison of different approaches to interaction would lead to a better application. They are named based on a short concise description of actions taken to trigger an event. These interaction methods are important to designing an effective UI for cyber-security tools because they provide a way for users to intuitively understand and navigate through the application. The central purpose behind hacking in VR is to visualize vulnerable hosts or nodes and apply some exploit to them in a fast and efficient way. These three main approaches that do this are as follows:

1) Hover-Touch, this method requires no button presses. The touch is detected when a 3D cursor falls beyond a certain threshold. This is determined by a velocity and position of the cursor to verify that it has passed through a target object.
2) Point-and-click, this method is how most UI in VR achieve interaction. It works similar to a projector laser pen, where a ray is cast onto a flat plane and triggers an element.
3) Hover- click, this method is a variant of the Hover-Touch, and thus it only detects when the user is hovering over an element. A click from the button on a controller is required to fire off an event.

These methods allow the user to interact with the same set of data and commands but differed drastically in the action from the user that was required to activate a command or event.

The hypothesis behind these methods was that a clicking action seemed to be more inaccurate than a touch action (the hover-touch method).

Upon collecting the data and performing the analysis, the overall data suggests that the hover-click method performed better than the hover-touch method. This was since hover-touch require more travel for the cursor, which when factoring in the noise caused by a human positioning their finger would be more likely to cause error. The human element does not differ from subject-to-subject in this case because the actions recorded can be generalized as a translation of the cursor with some slight noise applied. Although one can consider human factors such as age, fine motor controls, height, length of arm or fingers, and many others, this study focuses more on the general terms of movement that can be comparable to the simple translation. Conceptually, the data gathered would not deviate too much from a much larger aggregate pool since it is focused on only basic interactions which don’t rely on human behavior but more on the simple movement of a finger or hand shortly from one point to another.

2. VIRTUAL REALITY BACKGROUND

VR has had a long history, one just about as old as photography itself [1]. Essentially, the technology consists of creating visuals that contain 3D-depth coupled with a synthetic world that surrounds the user. Stereoscopic visuals have been developed with photographs since the 1830s, as seen in Figure 2, and for a hundred years until after the great depression, it remained as simply 3D visuals. In the 1930s, Edward Link developed the first flight simulator (Figure 3) [2].

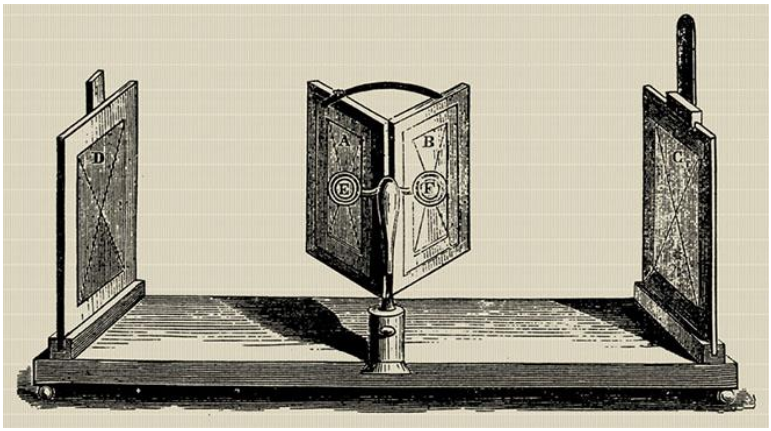


Figure 2. The oldest form of VR was a stereoscopic image viewer [26].

Figure 3. An immersive flight simulation playing back pre-recorded visuals to give the user a sense of depth [26].

This was used to train over 500,000 pilots during World War 2. It was called the Link Trainer, and although TVs were not yet widely used, it employed a sense of immersion unlike anything else before. Pilots would use it to familiarize themselves with controls and instruments before flying a real plane, which would allow them to train faster and safer before jumping into a real one.

The 1950s would feature camera driven 3D motion pictures as cameras became more advanced and capable of recording and playing back 3D visuals [1]. Eventually the 60s, allowed for head mounted displays to be possible. Although these were still crude and weren't designed for interactive applications but allowed for real-time feeds of dangerous situations.

By the 1980s, or the golden era of arcades and consoles, VR finally had enough computing power to run as interactive games. The first use of the term "Virtual Reality" was by Jaron Lanier, founder of the Visual Programming Lab (VPL) [3]. In this era, they would develop headsets and gloves (as seen in Figure 4) that could display very basic 3D graphics at very low frame rates, and thus would not become a widespread consumer product, but conceptually way ahead of its time.

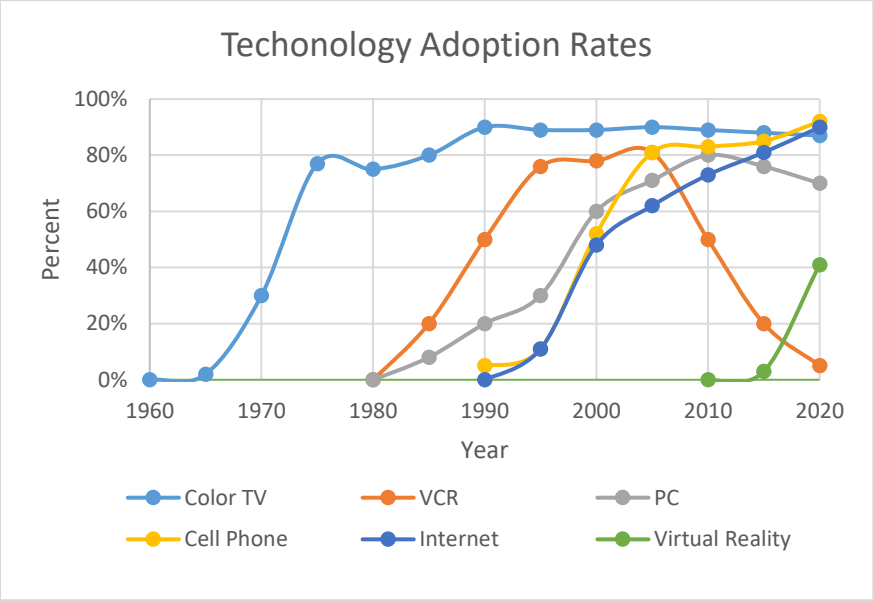
Figure 4. The first VR headsets were developed by the Visual Programming Lab [26].

The 90s, brought more interactive consoles with more powerful graphics processing to homes all around the world. With them, came the first consumer level VR such as from Sega and Nintendo [4][5]. Although neither ran truly 3D games, the stereoscopic aspect, frame rate, and concept was more alive than ever. However, the failure to sell these concepts to consumers caused VR to become abandoned by major console developers. After the Virtual Boy from Nintendo, a failed attempt in consumer level VR, the industry would lay dormant for over 15 years.

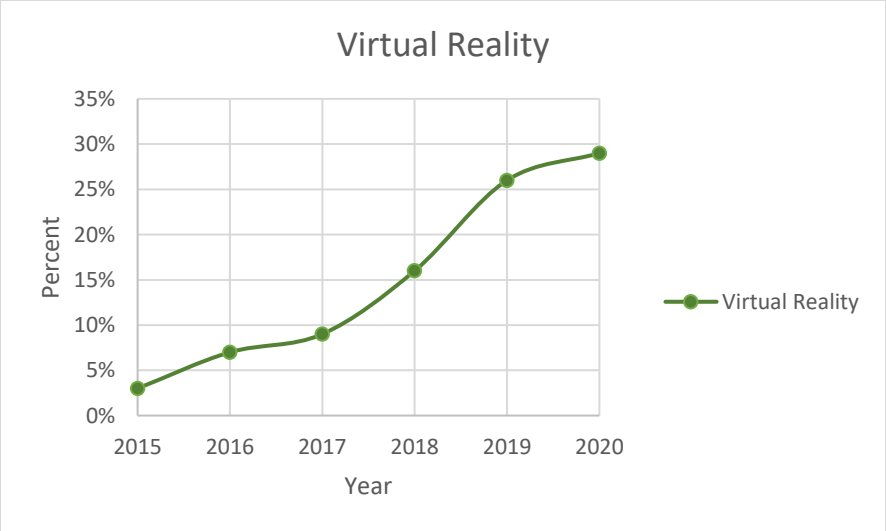
Eventually, the mobile phones market took off, driving smaller and more powerful computers to be developed [30]. Mobile phones are small enough to fit in a device that could be mounted on the user's head, their sensors would enable fully tracked movement and rotation. Eventually Palmer Lucky teamed up with Jon Carmack and raced to be the first people to announce the Oculus Rift in 2012[6]. Soon after it was bought by Facebook and would go into full production. Around this same time Valve Corporation also was developing their own version of the hardware internally [7], solving the room-scale tracking problem in a unique way by using their "Lighthouse" technology instead of the Oculus's infrared camera tracking system. By 2016, both Facebook and Valve released the first generation of true VR headsets. Since then, numerous other companies have followed the path to creating VR devices and new methods of tracking the player at room scale have been developed.

In the present, VR has not been widely adopted by the mainstream market. Up until the release of the Oculus Rift and HTC Vive in 2016, it was mostly a subject of academic research and niche markets. Currently, VR makes up less than 1% of PC gamers that use Steam, the current leading store for PC games [31]. However, in other industries such as health care, engineering, retail, and real estate make almost half the market [32]. This means that VR will

soon become more ubiquitous and utilized for more than video games as an interactive display technology, much like televisions, PCs, and mobile phones before it (Figure 5) [23].



(a) Technology adoption rates for major breakthrough in home electronics since 1960 [23].



(b) Virtual reality adoption since 2015 [23].

Figure 5. Technology adoption rates of major electronic media products in U.S. Households [23].

The adoption rate of VR has also not been great since it is costly and can be difficult to use [35]. Ultimately it is believed to be the future of computing, and a new way to experience

immersion in real-time graphics [24]. In contrast to traditional 2D based UI, modern VR can apply 3D graphics in a way that can make it even more intuitive to use due to the nature of it allowing a user to utilize a small screen and expand the interaction to a near limitless volume.

While VR is known as a cutting edge in graphics and hardware technology, it is still dwarfed by phones, game consoles, and personal computers in terms of sales. VR is regarded mainly as a technology for gaming enthusiasts, whereas the casual consumer might not be so inclined to use it to fill their everyday needs. VR combined with AR does have many commercial or industrial applications. Applications that require immersion within the information being displayed might be the driving factor behind VR adoption with in the larger software market.

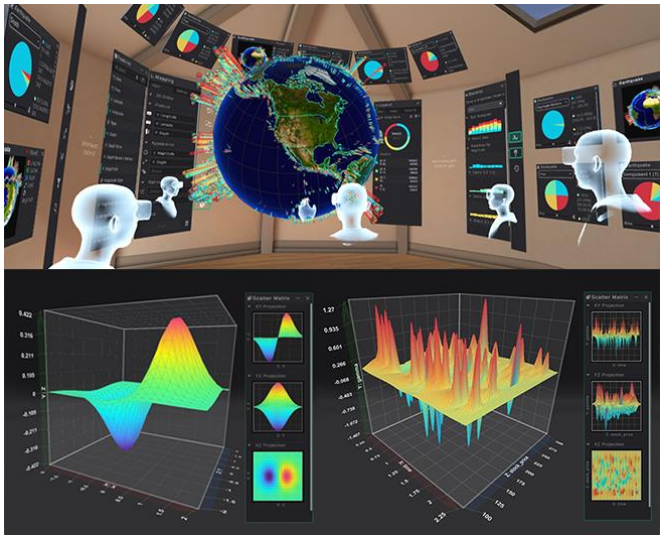


Figure 6. Virtualitics is a data analytics platform developed by Caltech and NASA for analyzing data in 3D space [29].

Many commercial applications of VR and AR exist today to help researchers and expert industries visualize data. One such application, developed by Caltech and NASA is called Virtualitics [29]. This application addresses problems with 2D visualization of data where turning a graph into a 3D environment does not make the data much more intuitive due to the

lack of depth. Combine 3D graphs or geographic data plotted on a 3D map with VR, and it becomes much easier to understand why a VR application displaying data in 3D space makes much more sense.

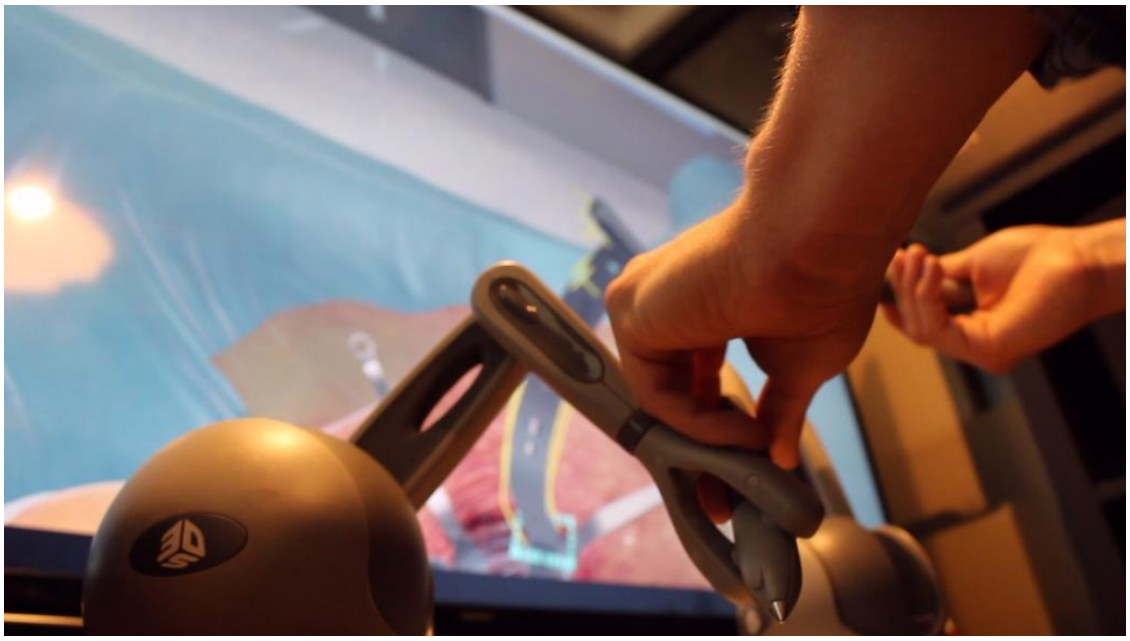


Figure 7. Fundamental VR is a surgery application for training surgeons [27].

Another key industry in VR is the health care industry. Certain skills can be trained just as effectively in VR as in real life but in so much more detail. For example, Fundamental VR is such an application that can train surgeons on how to perform surgery [27]. On top of having hands on experience with the tools, the trainee can be given instructions in a fully interactive simulation and refine their surgery performing skills. Not only is this an effective and efficient way to train surgeons, but a lot of data can also be collected on their performance.

The combination of VR and information technology can produce an application that takes advantage of the medium by immersing the viewer into a visual and immersive representation of the cyber world. This application could help drive the VR industry into the cybersecurity market and allow IT professionals to utilize the tool to manage and enforce security across their networks.

3. MOTIVATION & METHODS

The main drive behind this research is to explore different methods for developing a UI framework suitable for conducting work in VR. A cyber-security research software is ideal because of its involvement with data visualization and interaction. Often in movies and gaming media, the cyber world is portrayed in a 3D space. This is likely due to the fact that this data is so complex, that explaining it visually on the screen makes little sense to the viewer. This concept is a staple in the reason why a VR application that can accurately and functionally represent the cyber world made sense to drive the development of a UI framework.

This primary objective of this research is to develop a framework for good VR application design in the field of cybersecurity. For an application to be designed in VR, the developer must consider the purpose of using VR for the app. How does this application improve with the addition of VR? An application that is difficult to use in VR but is simple on a flat or mobile screen is unnecessarily complicated. However, an application that can fully leverage VR dimensions can evolve into a new kind of application. These tools, for example can work like an operating table for a surgeon, a multi-instrument music tool for a composer, or an immersive 3D sketchbook for aspiring artists [8]. The underlying framework is designed to represent some form of complex data, combined with intuitive and immersive controls that can help the user better manage tasks from a high-level perspective. Ultimately, this could enable non-cybersecurity experts to command automated hacking tools and understand the data analysis and decision-making process, this is also similar to how other works have developed tools to teach students using virtual learning environments. In a classroom example, students are given a virtual sandbox environment to experiment with the execution of cybersecurity exploits and intrusion detection [9]. Educational VR applications can place the students in an environment that is

related to the subject. This allows the student to become fully immersed in an experience like how a field trip can produce stronger memories than that of sitting in a classroom. The VR application should not only enable a user to interact with the data, it should allow them to become immersed in it.

3.1. Research Goal

Project Nighthawk is a virtual reality based cyber-attack simulation. It was created specifically to support the ideas represented in this research. It is a collection of tools for visualizing graphs, interacting with information in both networks and geographic representations, and interfacing with the cyber-security research sandbox environments in Kali OS. The major focus of this research is about designing an effective UI framework and studying the different properties of each unique approach. Cyber-security research using the Kali OS, has been a widely used tool to educate students on how cyber-attacks are carried out [10]. Kali OS was created to host a set of tools in a laboratory environment. These tools allow the user to simulate a hacking scenario. Similar to how a flight simulator teaches a pilot how to fly, the specialized OS allows a network administrator to understand how cyber-attacks work and how to prevent them. Similar to Nighthawk, the combination of a VR interface on the front-end and the OS back-end can simulate a cyber-attack scenario. This model for creating cyber-attack simulations would inspire the creation of demonstrations and ways to visualize and navigate through cyberspace in VR, modeling the data after real world systems.

More specifically, the project models computers and servers on a network represented by nodes and their connections to one another. This allows the user to see how each node accesses the internet via some sort of router or gateway. This mimics how real-world system administrators look at network graphs and monitor for any signs of intrusion through the network

traffic. Such as in many cyber-security classrooms, students would experiment with different scripts and programs to simulate intrusion [10], they could use this tool as a way of conceptually understanding how a cyber-attack is carried out.

The project also showcases some of the UI systems used for viewing data related to each node, planning, and carrying out an attack on the nodes. The state of each node is modeled and simulated based on the procedures carried out in the Kali OS hacking simulations. Although Nighthawk simulates hacking, it does not actually contain any malicious code, or hacking tools itself. It is simply a client, with a library of VR tools for interfacing with a network infrastructure that allows the user to access some of the tools in Kali that can simulate the attacks. The Nighthawk system connects to the operating system via a local network, which allows the VR system to access commands that can handle instructions for various tools on Kali.

While the primary goal has always been to design better UI and data representations in VR, the adaptation of models for cyber-security research has helped dramatically in designing the system. The data analysis and conclusion of this project will be based on what has been implemented and looking at why those choices fit within the context.

3.2. Challenges

The biggest challenge that the project faced was to design an open-ended system that can do what many flat screen tools have already done fluidly. A major VR application that can hack another computer is not currently known to exist at the time of researching for this paper. This meant that the concept of hacking would have to take inspiration from many of the previous depictions of hacking in media such as films, TV, and video games. The study utilized Metasploit and Kali OS to carry out simulated hacking on virtual machines. The research is not designed as a usable cyber-attacking tool, and rather provide more interactive ways use hacking

tools. It takes an in-depth look at what were the best practices or approaches to developing an intuitive and useful UI to working with the cyber-security research tools.

The project would also face some challenge when implementing tools that visualize data in 3D space. Many iterations of visualizing nodes on a network would be built from the ground up to provide a visualization that the user could interact with and understand how the information is represented. An example would be the implementation of Wi-Fi routers plotted in 3D space. How would this information be helpful in VR, and how would the user be able to understand and interact with what they are seeing? The reasoning then would be, because a geographic problem presented itself, and thus the solution might require the user to navigate or simulate the interactions in a 3D geographic map.

3.3. Project Nighthawk: The VR Cyber-Security Research Lab

This section will review central question of this study, and the steps taken to find the answer. The construction and composition of the project is covered in the related works section. The tools in Nighthawk consists of various VR tools available for free in Unity 3D which all extend from Steam VR which is the primary interface between the game engine and the hardware.

The main portion of the project will be covered under the implementation section, where this paper takes a look at the different iterations and methodologies used in testing out different components and studying their properties, advantages, and disadvantages. Finally, the last sub-section lightly touches on some of the movement data gathered. This is to help illustrate how button presses and interaction requirements can cause different behaviors on the user's positional accuracy and margin of error in triggering interactive objects.

3.4. Hypothesis

This project's primary goal is to investigate a framework for UI to be designed in VR to support a cyber-security research application. The main idea centralizes around the tools used to build 3D spatial interactive systems. The goal is to take advantage of the VR space to showcase complex data and provide the user with a way of interacting with it. In addition to this, a developer would want to leverage the VR space to utilize its functionality that would otherwise not exist in any other medium, such as a web page, desktop, mobile app, or a command terminal. The secondary objective of the study was to find different scenarios where these tools can be applied. The main focus of this research are the 3 different patterns for UI development. To review the 3 patterns, they are the Hover-Touch, Point-and-Click, and Hover-Click methods. The following are the hypothesis made and which are assessed through this work:

1) While in VR, the best way to interact with the interface is to simulate reality. Therefore, the hover-touch method, with its intuitive hologram button press interface, should allow for the best experience and response.
2) The Point-and-Click is too similar to 2D screen UI, and due to its shotgun like affect amplified at longer ranges of interaction, should be the worse.
3) The Hover-Click method should lie somewhere in the middle of the previous two methods in terms of usefulness and accuracy. While it still requires a click to trigger, the close proximity to an item should help avoid accuracy issues. In addition, this method may lose some accuracy due to the requirement to click on a button with a controller in hand.

These systems are the core mechanics to interacting with the virtual world. They are listed here each as advantages that the user will have over traditional flat screen monitors and

traditional input methods. In order to hack a system in VR, the user must be able to see the information and react to changes within the system easily.

To better understand these interaction methods in VR, the research takes an iterative approach to developing different tools that would be used in a cyber-security related setting. Some of these tools are inspired by cyber-attack scenarios such as acting as a simulated hacker poking into a secured network and scanning different devices on a network to find security holes. As the approaches go from simulation to scenario driven, the research shows how abstract methods for modeling a system can be combined in different ways to create a functioning and intuitive VR application.

4. IMPLEMENTATION

This section reviews the process behind designing the systems and implementing the tools and demonstrations to evaluate how the different UI packages would work in tandem to produce an application that the user could use to interact with a cyber-security research environment.

4.1. Tools

This section details works that have been done in the Unity 3D game engine to develop useful tools for creating VR experiences.

The Unity 3D game engine is used widely for VR development because it is a developer friendly platform for creating interactive content [11]. This engine was chosen mainly because of the programming language C# for scripting in-game behavior, creating user interfaces, and controlling some of the rendering and physics systems. This allows the developer to focus more on implementation of the system than the work of 3D rendering and management of input and UI components.

For the main input communication library, Steam VR was chosen since it has the most support of PC VR headsets. Developed by Valve initially for the HTC Vive, it is the baseline application for this project [12]. Currently it supports any device that runs on Steam VR, Oculus VR, and Windows MR. The Steam VR plugin itself, is a tool for the Unity game engine to allow developers to create content for the headset using APIs that can talk with the input devices such as the headset and controllers.

The Virtual Reality Tool Kit (VRTK) is a prototype tool used in the Unity engine to quickly develop and iterate on commonly used VR mechanics [13]. VRTK was originally developed to support development for multiple VR devices simultaneously. The features include

different movement mechanics such as teleporting, sliding, hand gestures, picking objects, UI interaction, object grabbing, object interaction, climbing, and many more common VR mechanics.

A forked version of the main project is used that supports Steam VR 2.X, a version of the library that has Valve Index support and remapping of controller inputs at runtime [14]. Due to dependency issues with Baroque UI, the original VRTK was used in the older branch, and a newer version was used in the newer branch that supports Hover UI.

The Baroque UI system, developed by the user Arigo on Github, it is another unity package that sits on top of the Steam VR library [15]. Much like touch screens on mobile phones or tablets, it mimics the user touching or coming in close proximity to a surface and interacting with those objects.

Figure 8. Star Trek Bridge Crew UI. The user is using a 2D cursor to steer the ship in VR [16].

This method was similar to methods used in a lot of games, such as Star Trek: Bridge Crew, which used a futuristic holographic interface where the player could interact with a screen projected on to a holographic plane in VR [16].

Hover UI creates a 3D cursor that can interact with a 3D UI system by touching the element. It can be used for any type of device as long as the inputs were provided for the UI cursor [17]. The UI system works by detecting intersection between a cursor and an interactive element, such as a button, slider, keyboard, or check box. This system factors in estimated velocity, proximity, and amount of time passed as the cursor intersected the element, to calculate and display when and where the interaction happened.

The Wrld API and Map Box API are used to aid in representing real world geographic locations. These libraries are designed for streaming geographic data into games and applications in real time to allow real world maps to be used at runtime [18, 19].

4.2. System Design

Initially the study looked at what kind of general components would be required in order to create a cyber-security focused user interface. This design was intended for a simulated cyber-attack to be possible. 3 key components were identified.

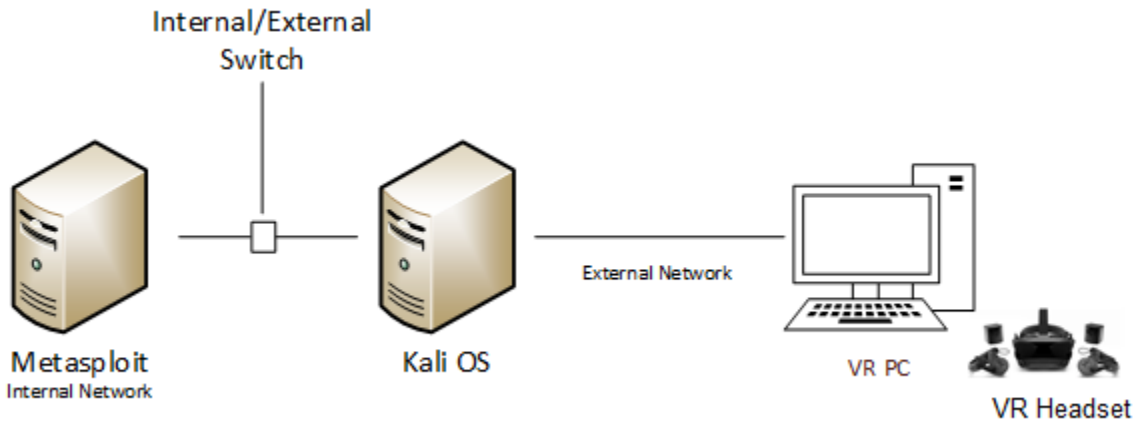


Figure 9. The client-server-host model for a cyber-attack simulation.

1.) The User PC, or VR capable device, is required to host the graphics and UI elements. This is the client in this system, it requires a server which will host the tools required to run simulated attacks on a vulnerable host.

2.) In this model a computer with the Kali Operating system, a cyber-security specialized system, was used to simulate a cyber-attack. The use of this system is based on a previous research project that allowed a blackboard system to automate and coordinate a cyber-attack. [20]

3.) The host system, which used Metasploitable, is designed to simulate the system being hacked [36]. Although this project is not focused directly on the cyber-attack portion, it instead will address the capabilities by demonstrating the framework required for the client and server to communicate in order to enable an exploit to be planted and later executed.

4.2.1. Network Communication

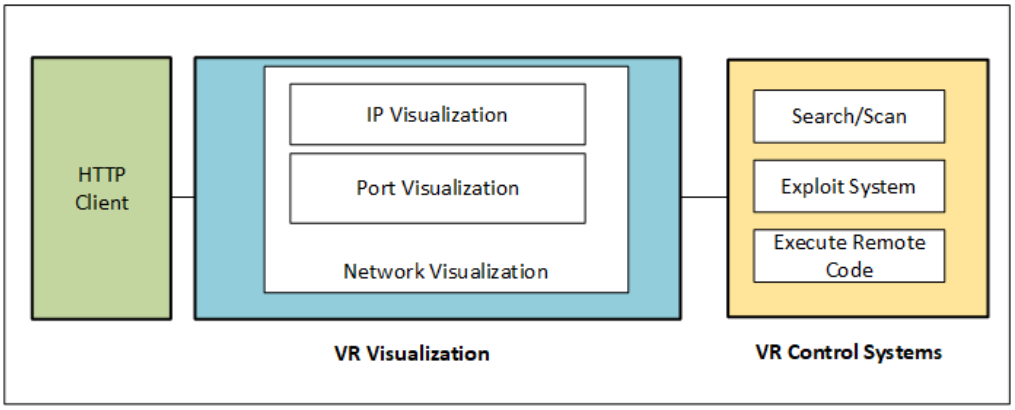


Figure 10. Network architecture diagram. The visualization system models and displays data, whereas the control systems allow the user to interact with the data.

After the initial design took place, a series of network level API calls were designed to allow the simulation system to communicate with Kali OS. This allows access to an internal network system via a set of commands that are designed and created within the VR system.

The possible commands are limited in this system as it was designed to follow the simple process of scanning for hosts, targeting a host, retrieving data from ports and services open, and executing an exploit. This exploit opens a “back door” into the system allowing for remote code

execution. The main purpose of this system is to infect many systems at once, or visually identify the most vulnerable systems and begin planting exploits in those systems first. An example of this system can be seen in Figure 11 The Lawnmower Man [34] where the main villain is searching every possible access point on the network to find a way to access the internet.

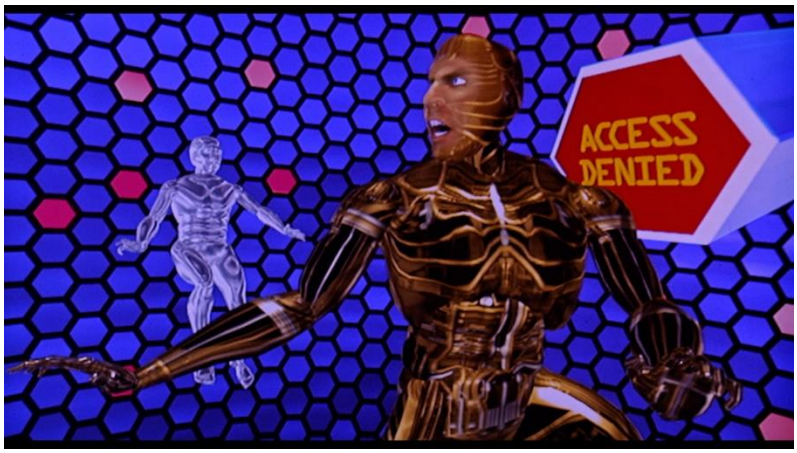


Figure 11. Virtual Reality hacking as depicted in The Lawnmower Man [34].

For the client visualization, Figure 12 shows some of the main components considered for the main action loop. The user would be able to use the control system to scan a network, visualize data, and make high-level decisions on how they would want to interact with the target host. Aggregate data could be shown in a summary of all the nodes scanned. Additional data on each node would be revealed in a sub-menu that would show which ports have been scanned, and what services were available to exploit on those nodes.

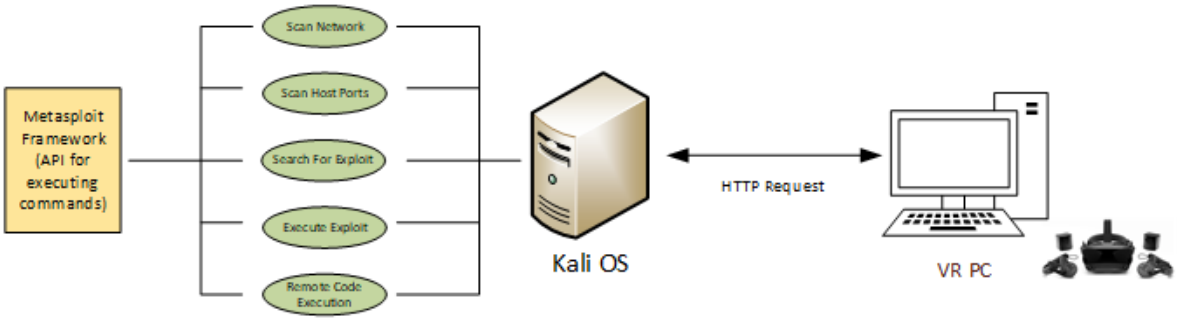


Figure 12. Network API design.

Note: The VR client uses this interface to command the Kali OS server to carry out a cyberattack.

The exploit system works by comparing each service, its version, and runtime environment with a database of known vulnerabilities as shown in Figure 13. In this way, it can quickly look up vulnerable services and create a list of possible exploits to run on the host. The user would then simply select an exploit to run, and the remote code execution would then be enabled.

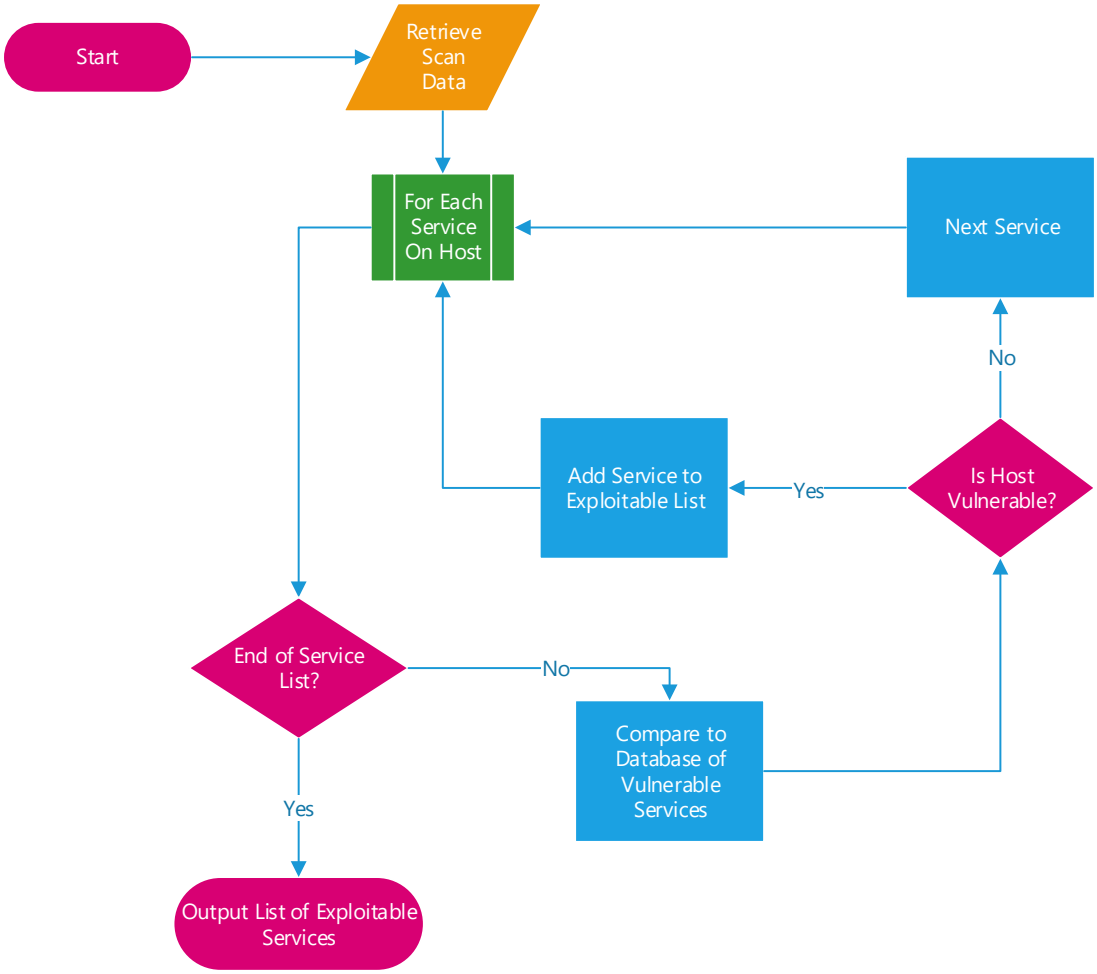


Figure 13. A flowchart of the exploit searching system.

4.2.2. Simulation and Scenarios

The data in this simulation must be abstracted in a hierarchy in order to be properly organized as an object representation when communicating between two endpoints. The host

network is a collection of devices and their relationships regarding the connectivity of the network. Usually these nodes would be behind a secured firewall, however in these visualizations, the firewalled nodes could have possible vulnerabilities where certain ports or services are exposed to the internet. This would allow an attacker to scan the network exposed and find vulnerabilities within the system that would get them into the network and behind the firewall.

Figure 14 shows the content of the network data consisting of a single node and its structure. It contains data relevant to the user to visualize and make decisions on whether the node is exploitable. In addition, this host data would also contain the ports and services that would be filtered out via a lookup table from the Metasploit database. The lookup table would then detail which exploits go with which service that's running.

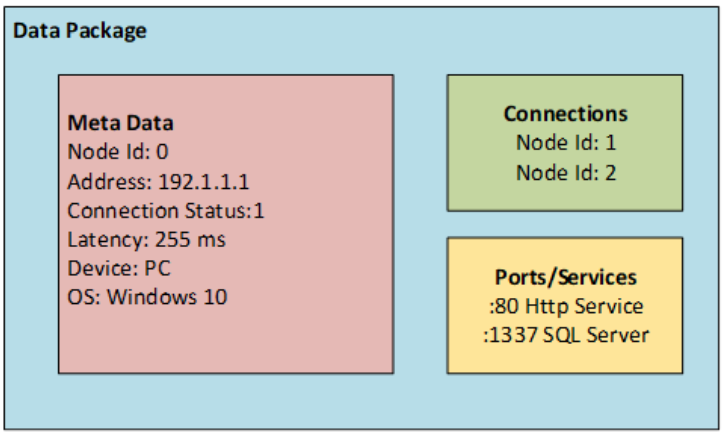


Figure 14. Network data package used to communicate between server and client.

The middleware between the client and server is responsible for formatting and sending this data to the client. It consists of an IP address, some status information, statistical information for the connectivity information, and the services it hosts.

4.3. UI Design and Development

This section discusses the different UI design approaches taken throughout this project. These methods use tools that were developed by others as a plugin for Unity. Each element is designed as significantly different approaches to interaction in a VR UI system. They enable the user to interface with data and execute commands such as scanning networks or executing exploits.

4.3.1. Hover-Touch Method

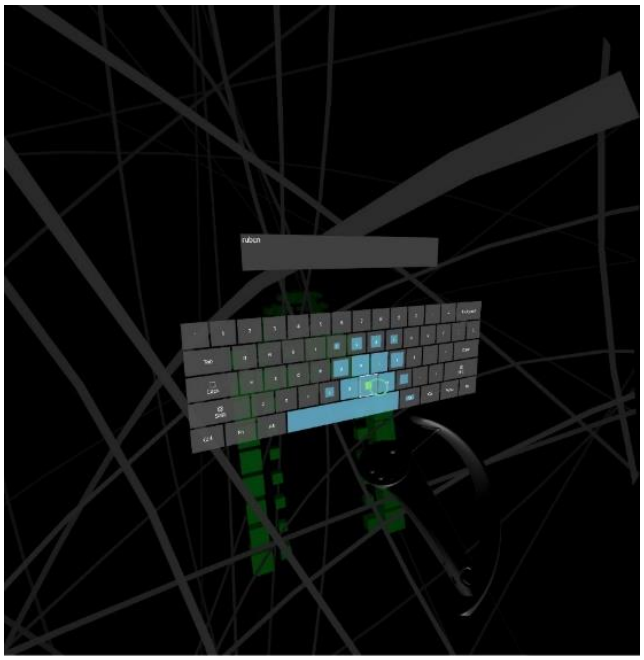


Figure 15. A holographic keyboard in front of the user's virtual controller.

This system works by tracking the position of a cursor that is parented to the position of the user's controller. In theory, all fingers can have individual cursors, but only 1 finger (the index) was implemented for this demo. The reasoning was because it would've taken more time and would not have worked as well to do since the fingers would mostly be fixed on the VR controller anyways. The cursor, much like a mouse acts as a positional guide and like Baroque's

proximity ray cast system. The main difference here is that it is always visible as a semi-transparent circle and that it triggers objects by lingering over an item and intersecting it.

For example, if the user were to use an on-screen keyboard, they could press the keys on the keyboard by simply tapping on it with their fingers, no additional presses required. However, the drawback here is that key presses in VR are more prone to error than in real life because it has no solid surfaces for the user to judge where their button presses lands. With practice and haptic feedback, this could build user intuition and improve over time. Thus, the hypothesis here is that it offers an overall better clustered-button solution than the previous two implementations. This method might be recommended for faster and more fluid interactions but should not be used as a button to confirm an input, such as a big submit button or navigation button, as the user could potentially misdirect their input into calling an action on a system with higher sensitivity.

4.3.1.1. The Interactive Graph

In Figure 16 the user shown interacting with a node on the network. Their left hand is represented as a menu for selecting operations that can be applied to the selected node. The right hand is used for selecting a node from distance, and interacting with the “hovering” UI in the foreground.

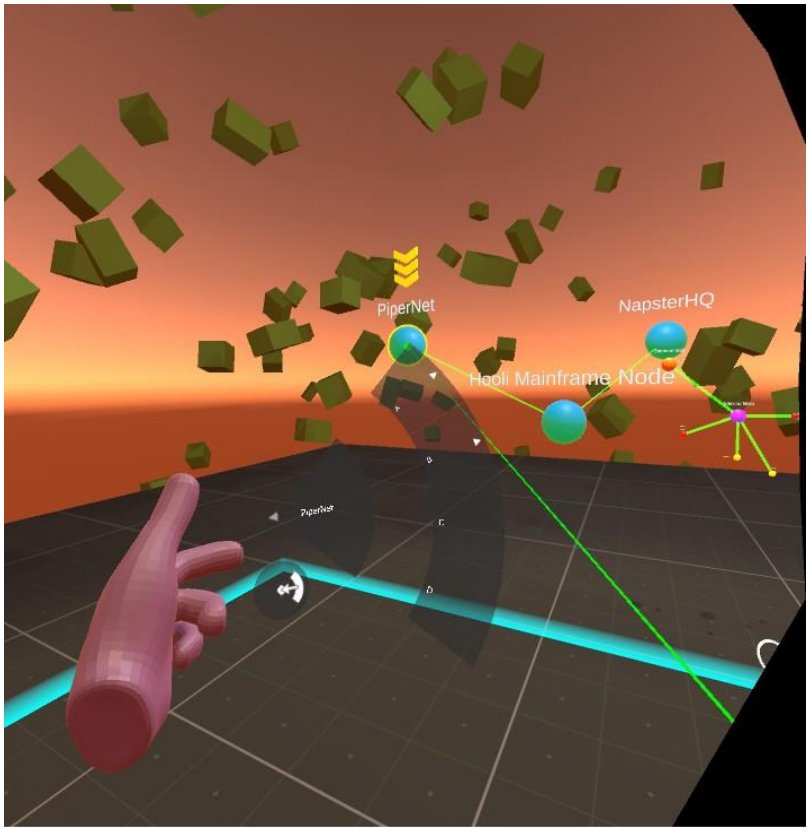


Figure 16. Assembly of all the UI and graph features results in a VR hacking experience.

This screenshot represents two of the interaction methods mentioned in this paper working in tandem. The first concept is the graph structure, where a set of nodes and their relationships are modeled in a connected graph. This structure allows the user to navigate through each gateway or network endpoint and scan the host for vulnerabilities. The second feature included is the Hover UI system. In this demo, it uses specifically the Hover Cast component, which is a radial UI system mounted on the user's hands.

Connecting the UI to the hands allows the user to adjust the position and orientation of the UI to whatever they desire. This also relieves any requirements to have to reposition the UI or to reposition the user in spatial relation to the UI. The UI also contains a navigation system like a web browser or mobile phone menu where screens are context driven. The context changes based on which options the user chooses.

Each state allows for transition to the next allowing different contextual options to be selected as shown in Figure 17. For example, the start node is select and scan a host on a network, from there, you can select a port, and whether that port is vulnerable or not, the user can proceed by attempting hack. The colors represent a node state, and the blue arrows represent a user action. The radial UI design in Figure 16 was chosen to provide a more intuitive menu navigation experience as the user is interacting on it with their hands. The UI can also resize if more elements are added to the list or even expand vertically if the options overflow the space allowed.

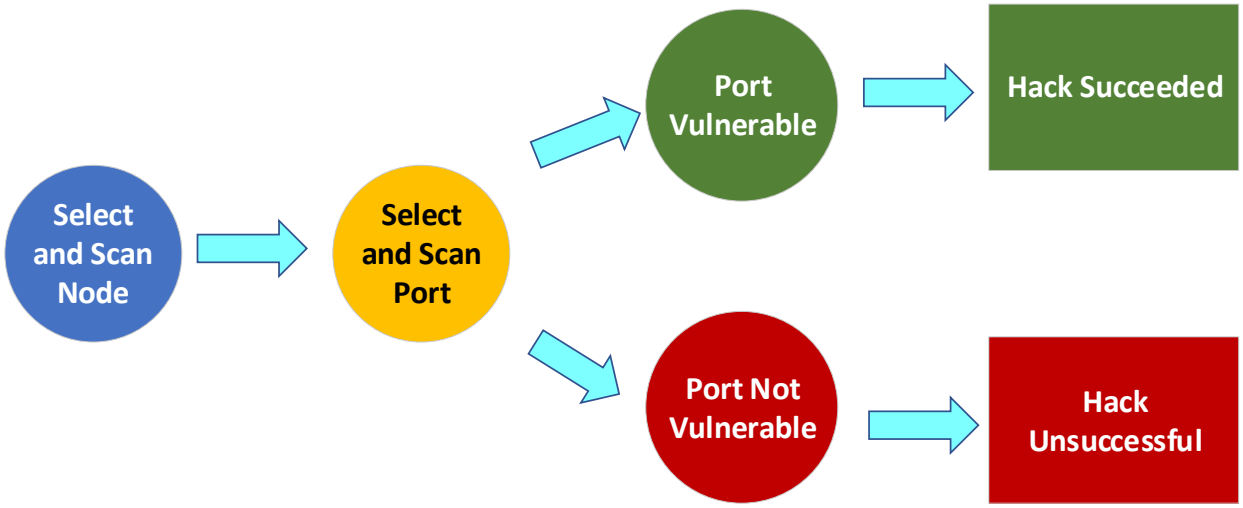


Figure 17. Flow of the simulated hacking process showing the different states of hacking a node.

Each node also has a set of states it can be in, as shown in Figure 18. Each of these states branch off scripted events created within the scenario of exploitable or non-exploitable systems. The simulation is designed in Unity as to mimic exploitation of actual nodes on a network. On a real network, the resulting information would be much more in detail and would require a lot of effort to set up more virtual machines to do the testing. However, as part of the design process, creating a simulation allowed the project to move ahead with an iterative design approach.

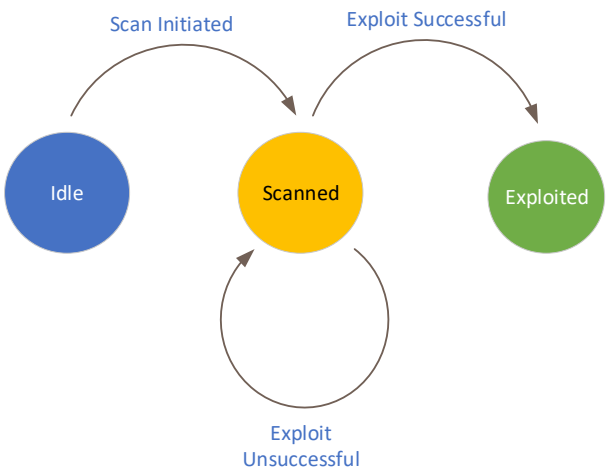


Figure 18. State transition diagram for each node in the network.

This model does not represent exactly the actions a hacker would take to exploit a system. Rather, it abstracts the actions taken into a semi-autonomous approach. The hacking system is represented as a series of circles and lines which allows the user to quickly understand where they are in the network. The system, then automates the process of searching and applying exploits autonomously on each node. This way, the user can explore many different branching paths through the network searching for nodes they can exploit along the way before carrying out the exploit.

4.3.2. Point-and-Click Method

A straight forward method that many interactive VR applications and games use is the point and click method for UI design. This method is common because of how it mimics flat screen interactions. A user would normally use a mouse, or touch screen to point on an item, and tap a button or surface to click on it. This method works by showing some sort of cursor (the green laser with a dot) that emits out a ray which intersects on a flat surface as seen in Figure 19 [33].

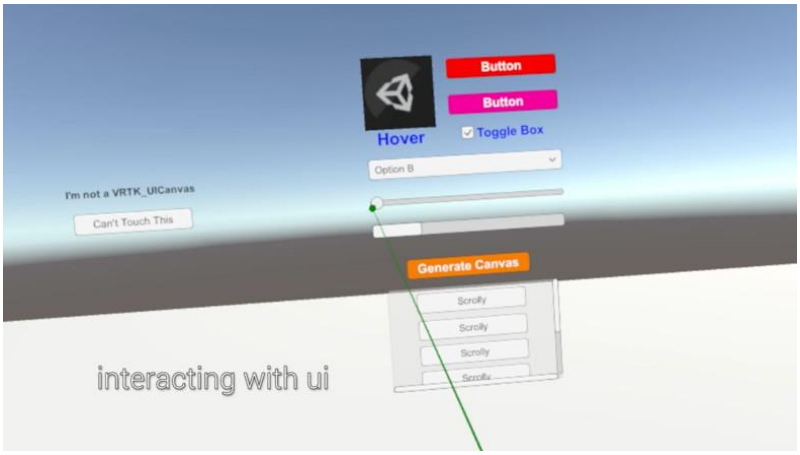


Figure 19. Virtual Reality Toolkit UI pointer example.

The input method requires the user to press the thumb pad and it would shoot out a ray. Then they would hover it over some element and press the trigger button to select that element. Variations can also include not requiring pressing a thumbpad and always ray casting to detect the UI surface, but the result is some sort of action to confirm the event.

While the pointer implementation is somewhat easy to do, especially with the built-in functionality of VRTK's pointer event system and Unity's existing canvas UI, its uses does have some limitations. For example, the pointing gesture has an added cost in angular stability when it comes to pointing from the hand or wrist to a screen just a few feet away. This becomes even more apparent with more buttons on the screen, which becomes harder to aim the ray onto because of the relative "closeness" in space to each other. The hand cannot fully stabilize itself since it is not in contact with a surface, and thus will cause some wobbling of the pointer, which is multiplied by the distance of the UI away from the user.

This interface is best for general UI interaction such as menu selection and navigating through in game scenes as will be discussed in detail later in the analysis and summary section. It is best avoided when trying to enter in fine input data or require a higher degree of input precision like a set of characters on a keyboard. Interactive media such as button prompts within

a 3D environment may work, but it is not recommended for something like a chat application. This method was implemented to test out some UI functionality but was mostly used to create scene transitions and object selection.

4.3.2.1. Nighthawk Menu

In order to illustrate an applied approach to show how the interaction methods would be used in the application, 5 conceptual examples were created to show how the Nighthawk project would work if it was used to hack other computers.

Figure 20 shows the tower configuration for modeling network nodes. This format could be combined with the geographic map to visualize sub-nodes or spatially close nodes in each stack. This model however, lacks the features of a connected graph. While useful for plotting central nodes on a map, the model doesn't show the relationship between each of the primary nodes.

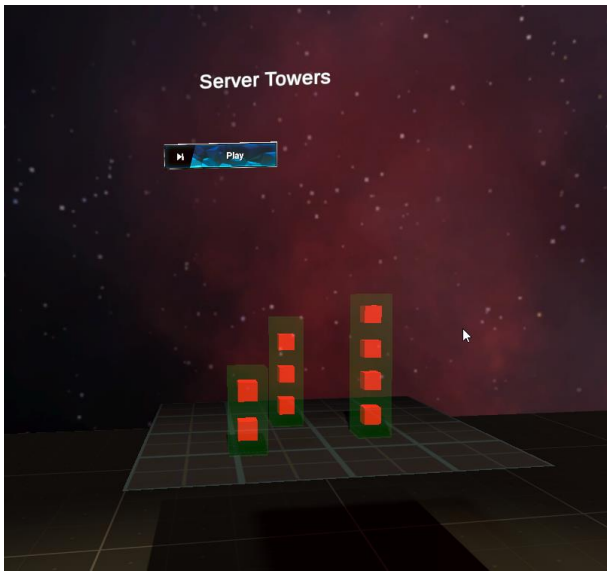


Figure 20. The network modeled as pillars that contain sub-nodes inside.

Geographic data visualization is one of the key features in this study represented by the model in Figure 21. The main question here was how network data can be displayed on 3D maps

that would be helpful to a user. The city scape example focuses on ranged wireless signals that can be plotted using latitude, longitude, and altitude. These coordinates can be correlated using a local x, y, z plot and a projection algorithm to match a point on the Earth's surface.

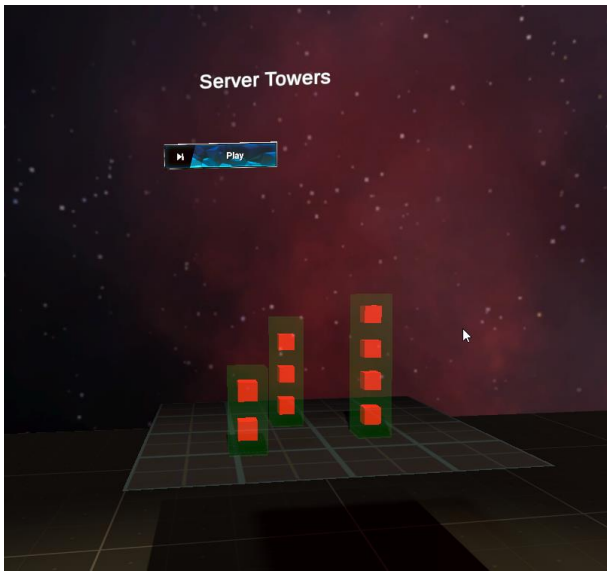


Figure 21. The network modeled as pillars that contain sub-nodes inside.

In Figure 22 instead of a local area map, the map is pulled back for a regional view. This example uses Map Box running in VR to show information on top of a terrain where a network node might exist (while a scene for this does exist, it does not work optimally since the Map Box tools used were not optimized for VR).

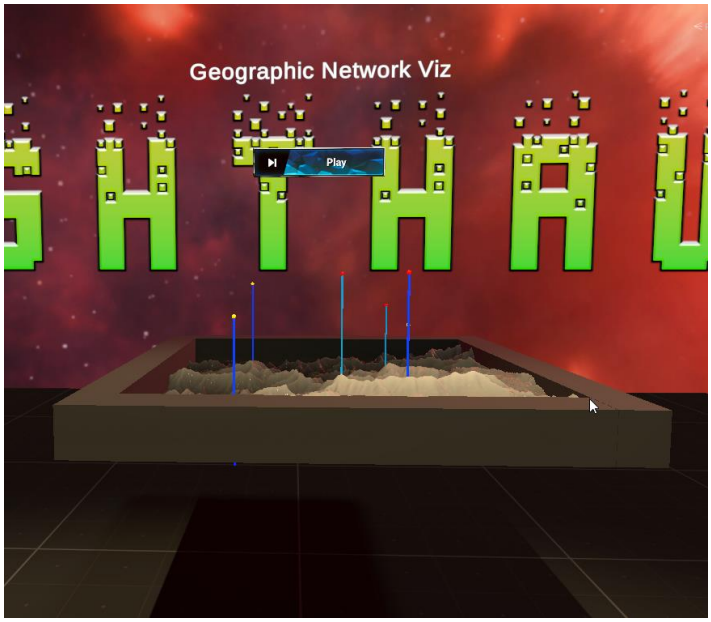


Figure 22. Regional scale for plotting IP address locations on a map, this example can plot nodes hundreds of kilometers apart.

Figure 23 and Figure 24 represents the node visualization systems. In Figure 23 the visualization is kept separate from the full interactive graph. This is done so that development on each of these features can focus atomically on the separate functions within the graph. For instance, the network visualization focuses more on the network overall, while the interactive graph focuses on each singular action taken on the nodes. This made the process of working on large-scale and small-scale interactions more manageable.

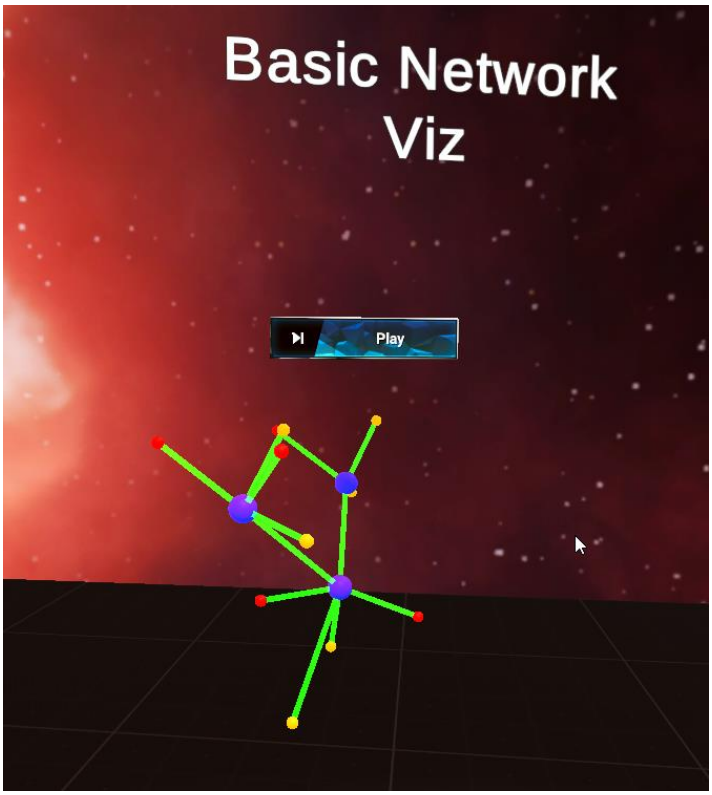


Figure 23. Network node graph visualization.

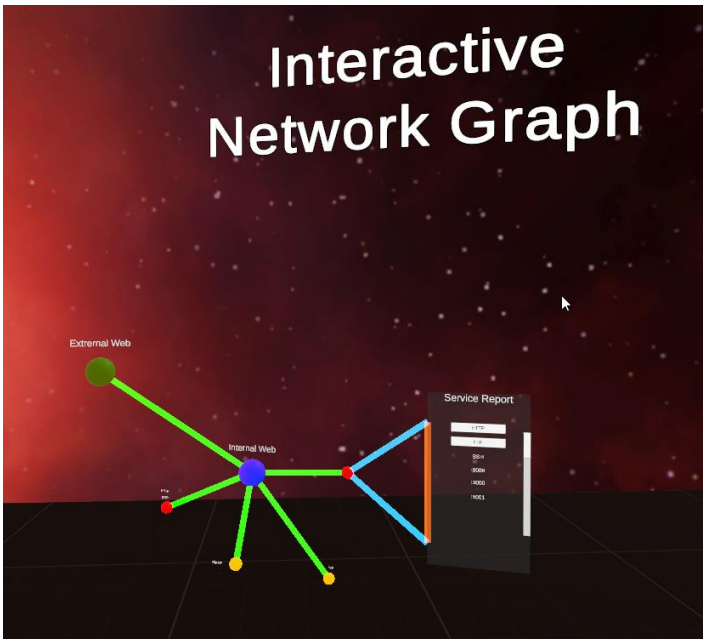


Figure 24. In the interactive graph the user can access data on each node by scanning it and executing exploits.

4.3.3. Hover-Click Method

This method uses Baroque, and is mainly for comparison against the hover-touch approach as it is designed for more accurate key press on a surface such as a keyboard or number pad. The difference here is that the user needs to press a button to trigger the input. The position of the finger and the element does not need to be precise, but the button that the user will be interacting with will be highlighted from a reasonable distance (about 1-5cm from the cursor). This method has some potential to work for either user preference or could be proven to be more usable than the hover-touch method.

Baroque is an open source UI package used to show interaction at a shorter range. It works similarly to the base Canvas UI interaction. In the basic button press demo as shown in Figure 25, the user hovers over a button and presses it to trigger an event which are keys on the key boards. In a selection demo, the initial trigger action causes a drop down to appear, and the user selects that drop down.

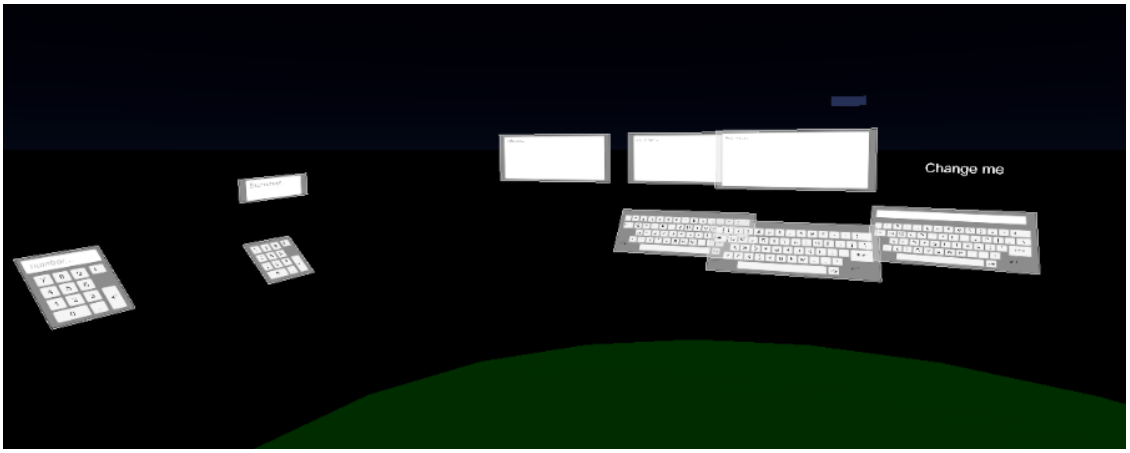


Figure 25. Keyboards and number pads in Baroque.

In contrast to the ray cast UI, this system allows the user to get up close to a button and perform a short ray cast to detect if a button is there to press. While this system has clear advantages with requiring the user to press buttons up close instead of further away, it has some

drawbacks since its range is limited. The hypothesis here is that the button press action can still interfere with the action being performed as any button presses causes slight movement in the controller. In a later UI system implementation, this study would look at a system that requires no button press and evaluate the functionality of that.

4.3.3.1. Searching System

To test the integration of the Baroque's hover-click function, a searching system was implemented. In VR, there will be many scenarios in which data will need to be entered, therefore some kind of virtual keyboard will be required. One such example is shown above where a search function for the hosts were required. The searching system was created to help input integers into the search box as seen in Figure 26 so that the user would have a way of navigating through the UI to find IP addresses. The text box is implemented using a parsing function that detects from left to right whether a user entered in a number or decimal point. A decimal point will trigger the position to start the next octet, overflows automatically insert a 255 for that octet, and a Backspace button will revert to the most recent left octet level. Once the numbers are entered, the Enter key is pressed to filter out the graph for the nodes. In addition, empty fields are treated as wild cards to simplify usage. The dialog box was also made available at any location if the user hits the trigger button to reposition the dialog box. This is a concept revisited later in the Hover UI system.

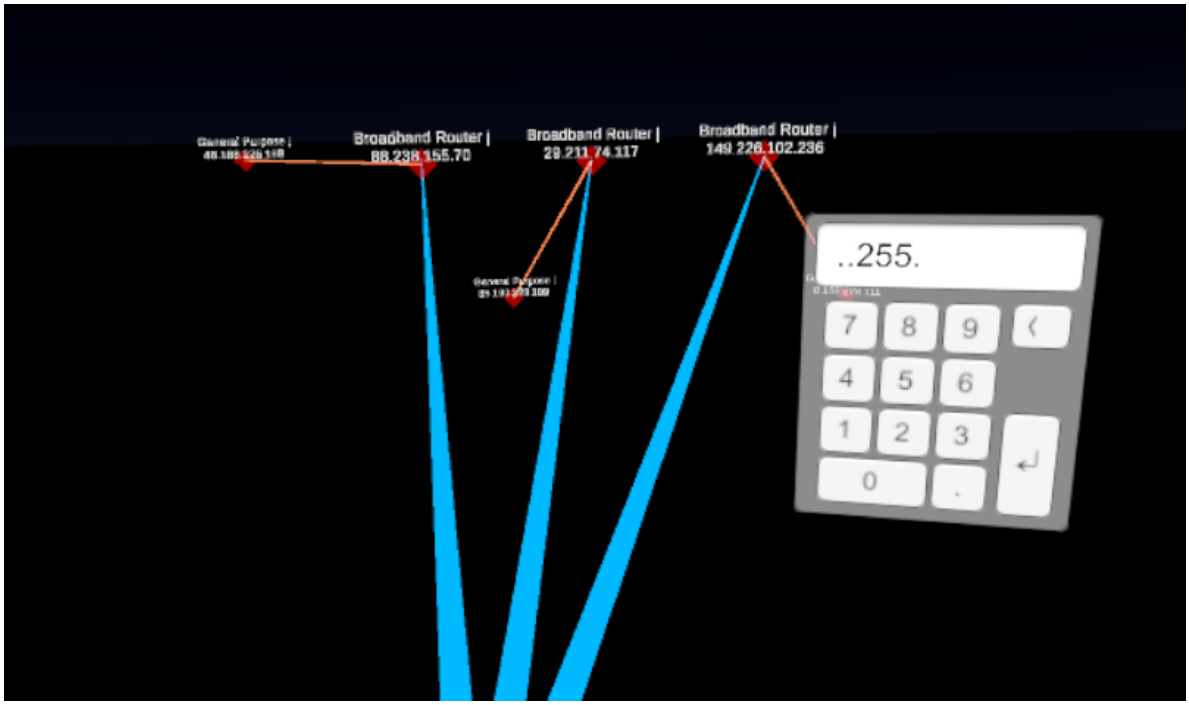


Figure 26. Searching UI system in more detail.

4.3.4. Accuracy Benchmark

The final section of implementation deals with button presses at various distances and types of buttons. Figure 27 shows a test environment of the point and click button system using the VRTK pointer tool. In total there are 3 different data collection settings to detect trigger events.

1) The hover cursor selection method. This method uses a cursor attached to a finger or controller to be guided into the trigger object and activate the event. This method requires no button press.
2) The Point and Click event trigger. This requires a 2-stage setup, step 1 is to hold the thumb on the Vive track pad to bring up the pointer and step 2 is to pull the trigger on the controller to click.

3) The final method is the hover-click method (Baroque UI) using a thumb press and trigger method. This method is like method 1, but instead requires a button press to trigger.

The data is captured at roughly 90fps and in 3 positional axes. It also doesn't capture rotational data since it is mainly concerning the position of a point being projected on a 2D plane which is the UI interface.

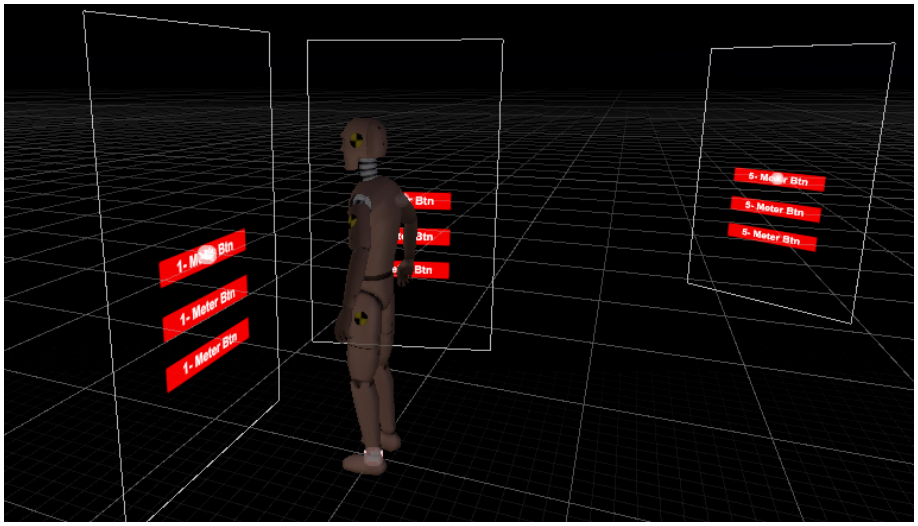


Figure 27. Test dummy for reference compared to UI panels.

The interface accuracy analysis in the following section will look at what advantages in object accuracy one can make from the different methods of selecting and activating a UI element. The aim of this experiment was to look at what data can be collected from a user selecting buttons from the UI. The data was not surveyed from many users, but from just 3 trials runs of the 3 different methods with a single user to get an impression of what this information can reveal about positionally tracked usage behavior at runtime.

4.4. Geographic Wireless Signal Display

In this example, the user can visualize wireless within a simulated geographic environment. This method is designed so that someone who has access to various signals within

that area can triangulate and triangulate a device that lies within the center. There is no functioning hacking being done for this portion of the project, but allows for a concept of what a geographic cyberspace overlay look like.

In Figure 28, the signal strength of Wi-Fi signal location is geographically mapped on the Wrld API which delivers spatial geometry data to the Unity engine. The Wrld API also can track features or custom markers that game developers spawn within the world. The markers shown here are spawned using an API call, their position is offset relative to the user's position which is placed at the center of the game world. The user does not move, but rather the system moves the game world around the user allowing for the entire planet Earth to be rendered around the user. For obvious performance reasons, the user can't see the entire world, so the renderer only loads and displays geometry for the surrounding environment with a cut-off limit set to whatever the camera frustrum uses.

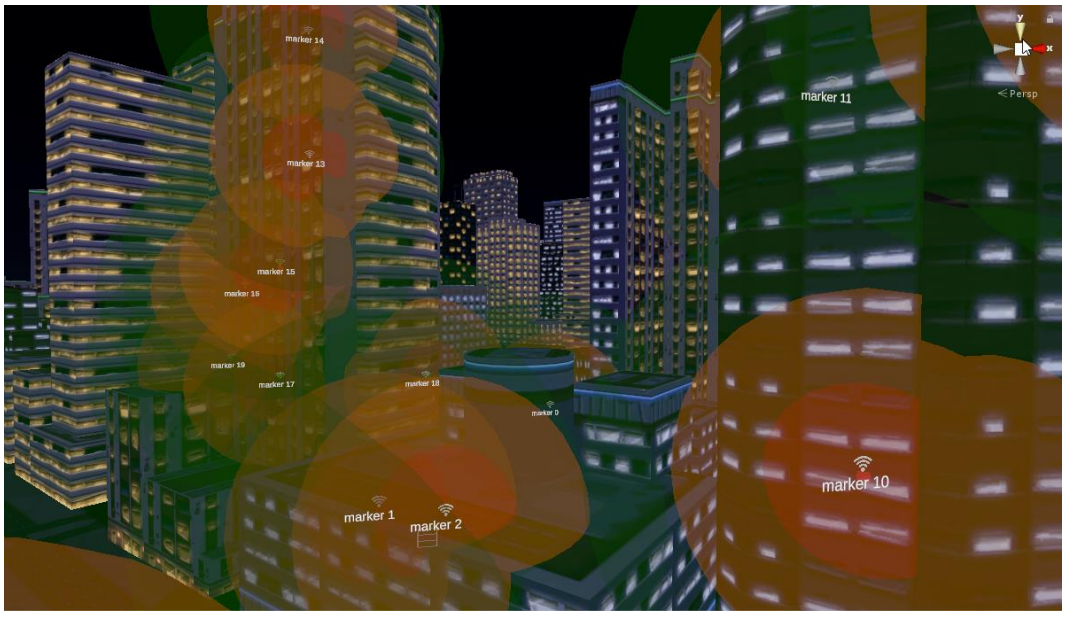


Figure 28. City wireless signal map visualization.

This example essentially plots a series of wi-fi signals set up in relation to a mystery device of unknown geographic location. Devices in a dense population area can generally see the

signal strength of surrounding devices. If the surrounding devices can be accurately pinpointed using latitude, longitude and altitude then the attacker could deduce the location of the device being tracked.

The demo geospatial plot data was manually generated using user input. The user would fly around the city deploying mock Wi-Fi routers or access points (AP as seen in Figure 29) and set the signal strength so that they visually triangulate using signal strength values and their locations. Ideally, there exists triangulation algorithms that can automate this process, and visually highlight target locations of the attacker, but this demonstration only shows a proof of concept of how the user would interact with the processed information.

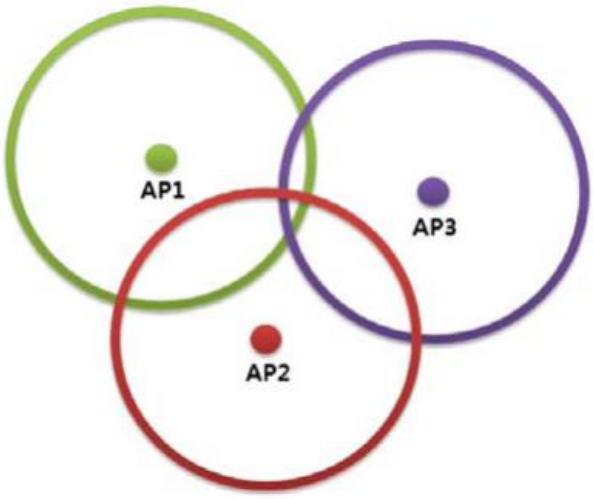


Figure 29. Overlapping circles on a 2D plane help to triangulate a central point [25].

In a fully integrated system, this concept could be applied to a drone or aerial vehicle to scan a specific geographic location for signals and plot them on a map. Likewise, a more efficient approach would be to use known markers within the world, such as a public Wi-Fi modem of known location, to scan and plot the locations of other routers within a certain range. This could build a map of other signals in the city and allow the user to visualize where the target might be. The target in this case would be a computer or device that the user can scan to find a

list of routers/broadcasters and their strength in relation to the target device. Ultimately, it would provide an idea of where the target device is located. This would allow for agents within the local area to take action such as arresting a criminal or investigating a crime.

4.5. Graph Design and Development

This section will go over the different approaches to designing the graph visualization system. It will also talk about the evolution of designing an effective graph for representing network data.

4.5.1. Bar Graph

Figure 30 shows a stacked set of servers containing either sub IP nodes that belong to a specific gateway, or services that can be accessed. Although this graph did not show connectivity, it evolved into the geographic visualization model that will be discussed later.

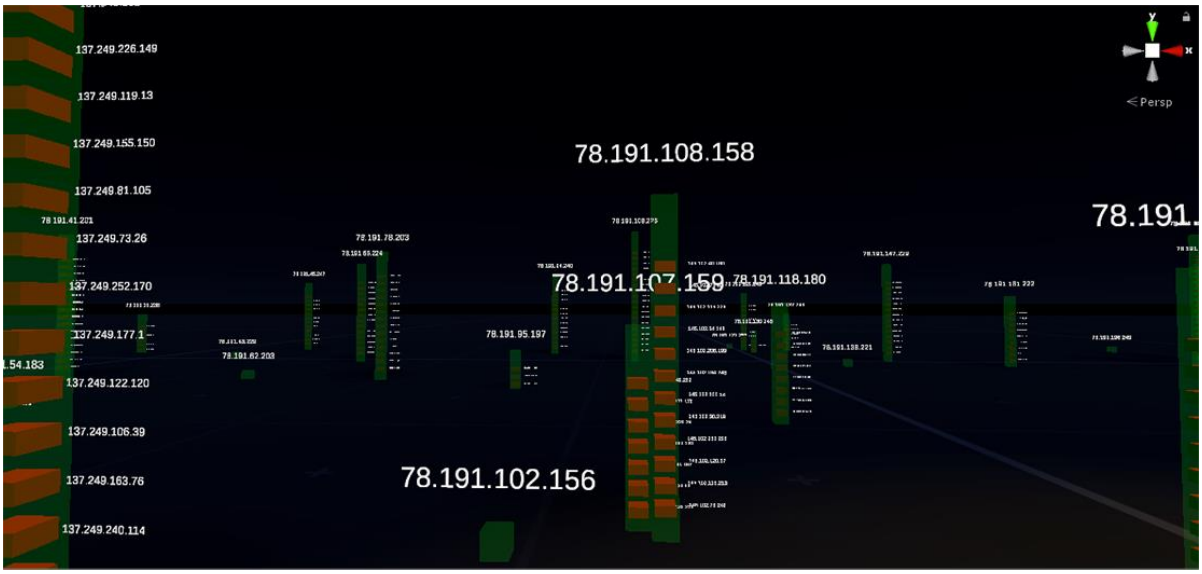


Figure 30. The network graph represented in a tower-like configuration.

This graph utilized the first iteration of the graph data model which contained parent and sub-nodes. The data structure consists of information related to the address of each node, and the services it contained. Test data was generated using Javascript and JSON was used as the storage

format for serializing the information. The server that generated the data would act as a simulation (the real meta data gathered from a network scan came from Kali OS) that would send to the VR session. JSON/REST API was used as the communication format since it was easily parsed into an object in C#.

In this iteration of the graph, the user could navigate through the environment using the directional pad found on the Vive controller. Interaction was limited to loading up the graph and navigating through it.

4.5.2. Undirected Connected Graph

In Figure 31 a different approach to data visualization is shown. It takes a more traditional graph approach to depict the network as an interconnected system. In Figure 31, all the nodes are connected at random making the graph very difficult to read. This is done as a naive attempt to model randomly generated data. A more structured approach is taken in Image 3 as the scope of the data was narrowed down.

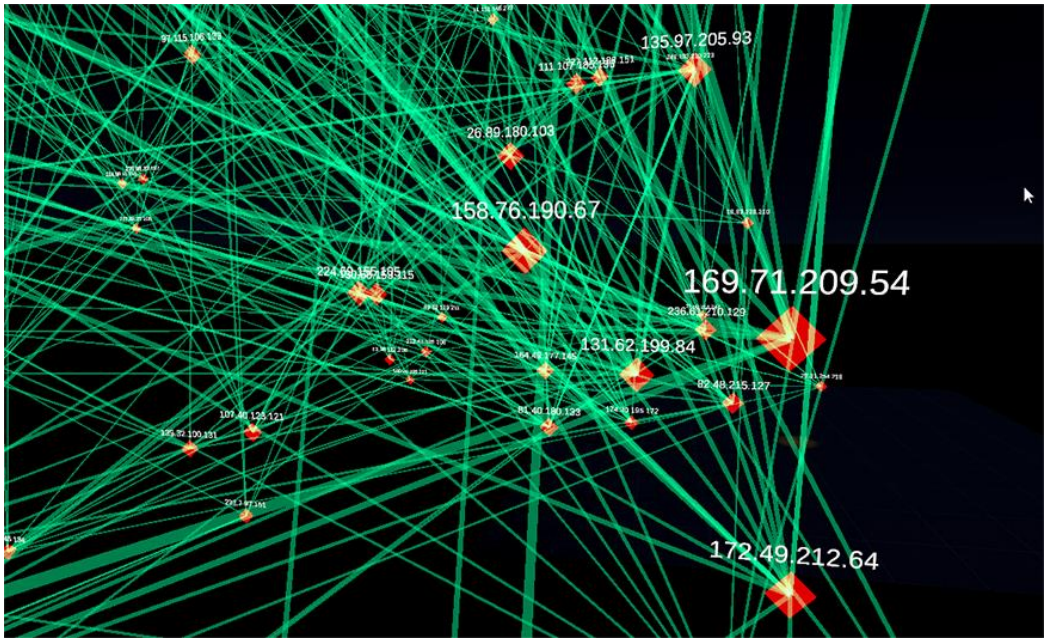


Figure 31. A graph generated from randomly connecting nodes.

The data still was usually confusing, with no way to discern which node connected to which, and what their relationships to each other. The real internet has a similar (but different) structure, due to how gateways are shared between multiple devices to connect to the internet. A router, internet service provider, or domain controller can be used to allow a device access to the internet. With the hierarchical organization of nodes in mind, the 3rd and 4th image graphs were produced.

4.5.3. Ordered Tree Graph

In Figure 32 and Figure 33, the ordered graph was created using a parent-child relationship connection to represent the hierarchy. This produced a visualization that made it easier to understand the relationship between each node. The inspiration for this visualization was taken from two primary sources: the NMap visualizer Figure 34, and the map of the internet Figure 35.

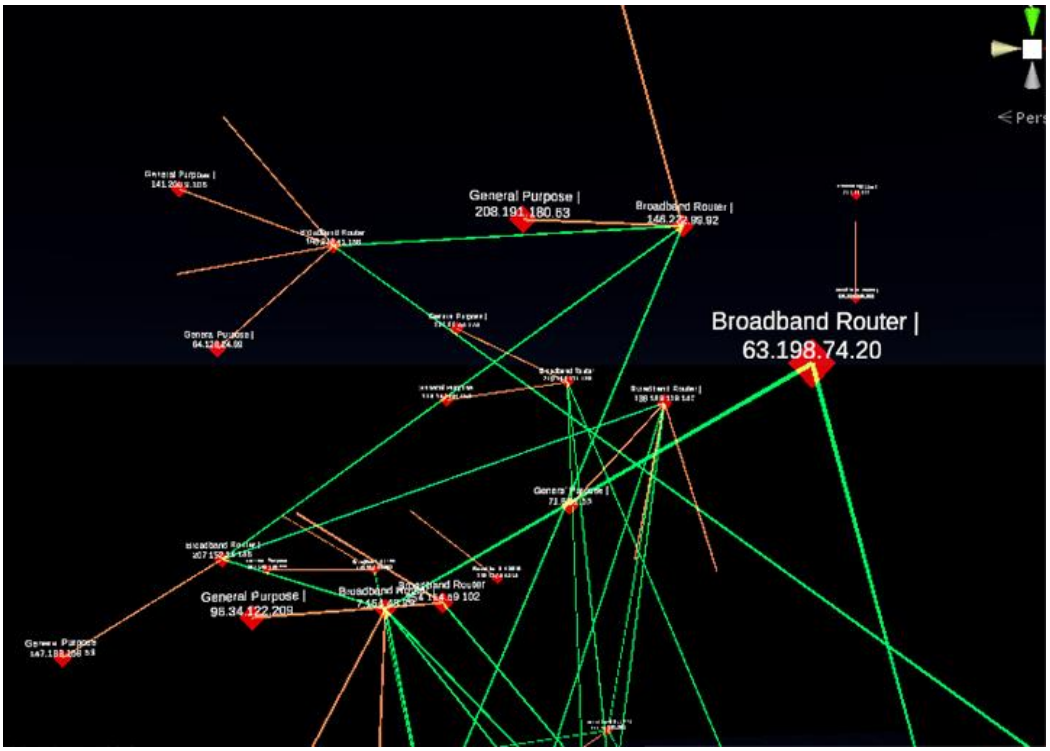


Figure 32. A structured version of the graph showing a hierarchical system.

Figure 33. A hierarchy based connected node graph using a radial algorithm to position in the child nodes in a more readable position.
Note: The graphs are ordered chronologically for different iterations taken.

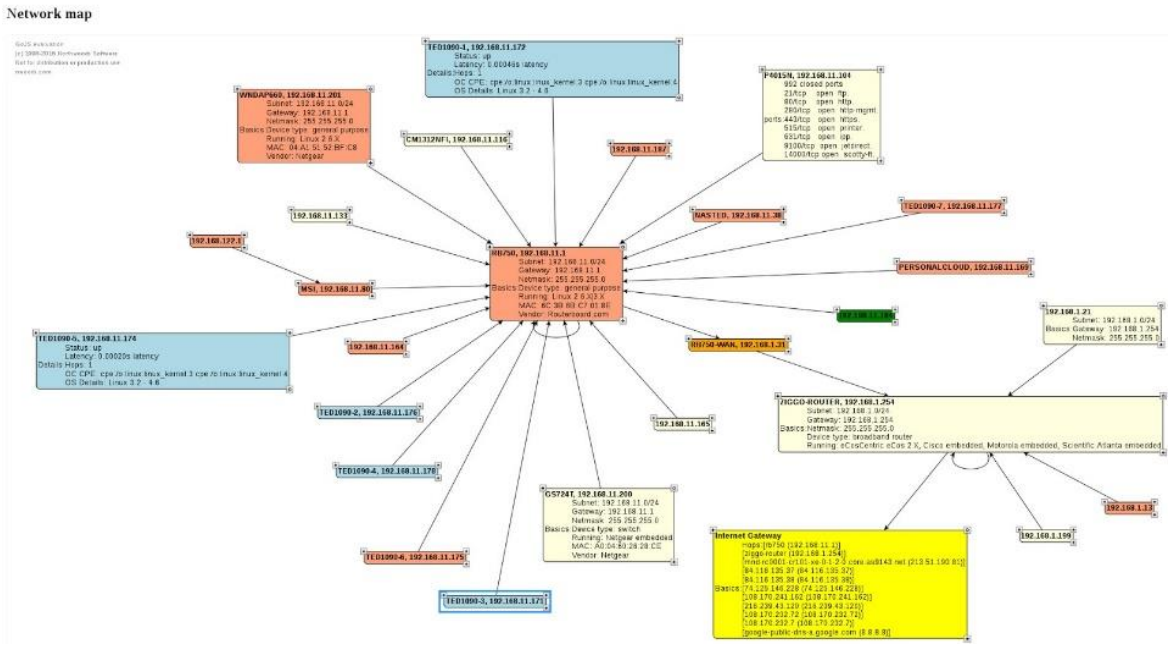


Figure 34. The Nmap graph visualization example [21].
Note: This would eventually make its way to how Nighthawk represents a data summary per node.

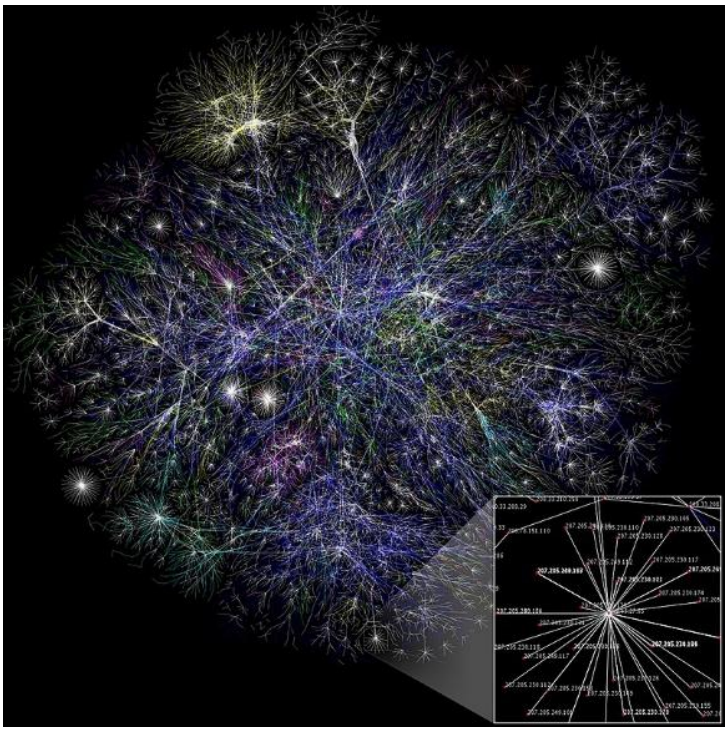


Figure 35. A map of the internet from the year 2003, the OPTE project [22].

The Nmap visualization is an open source project that shows a data about that node gathered from the Nmap program. A version of this program is included in the Kali OS server to scan data for a host. In addition, this means that the visualizer also shows data that is actually retrieved from a Nmap scan Figure 34. Note how the nodes are organized in a radial formation, with nodes connected to some master node, and each node you can expand or collapse additional data.

In Figure 35 below, is a map of the internet from the year 2003. It was most famous for showing how IP addresses are mapped across large regions. This cloud like graph was what ultimate led to the implementation of image 4. The main difference of the larger map seen in Figure 35 is that it shows a higher-level view of the network than in Figure 34. Combining these two concepts would result in the node map that Nighthawk uses, were relevant data stored is shown on a separate UI panel when selecting on a node to scan.

In Figure 32, nodes would only force in a certain direction as they simply subtract some value off a single X axis. This was somewhat counter intuitive to the user as closer inspection on any single area of node map, one would not be able to tell where they are in the graph.

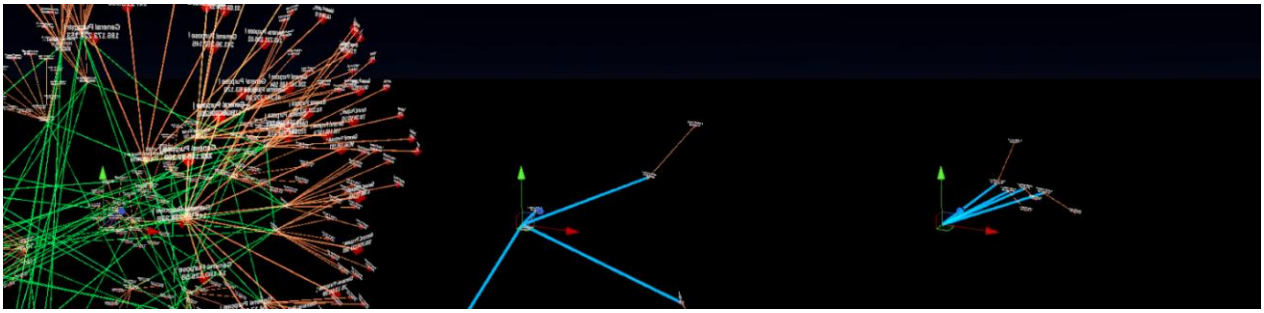


Figure 36. Animation of the search function from Project Nighthawk.

Thus, a rotation vector was applied for each child, where the vector would be the position of the parent offset by the center of all parent nodes. This rotation would then be multiplied to each child node to allow each child face away from the parent's center, using a radial offset

value position the child nodes outward. This gives each child node a compass-like look which allows the user to orient themselves in the space and also allows the “order” of each node to be separated. This would organize the parent nodes closer to the center, and child nodes farther away.

In addition to creating the graph, a search function was implemented, although not comprehensive (not including the ports and services) since the project is more focused on overall design, was able to allow user to quickly filter the nodes and see which part of the graph the nodes were originally from in a short animation.

5. ANALYSIS

This section will discuss the results of each of the implemented test scenarios. It will summarize the resulting application created, and what results each system yielded. Most notably, the UI motion data analysis section will compare the different UI implementation methods and discuss what situations each UI would be most suitable for in an application like an automated cyber-attack system.

5.1. Network Interface

The network interface implementation provided useful information on how the system overall would function. The example shown in Figure 37 is meant to illustrate how a client-side VR application would communicate with a platform that could carry out a cyber-attack using a list of scanned exploits. In addition, the investigation into building the proof-of-concept also allowed the researchers to gather more knowledge and experience carrying out simulated cyber-attacks using established cyber-attack research tools.

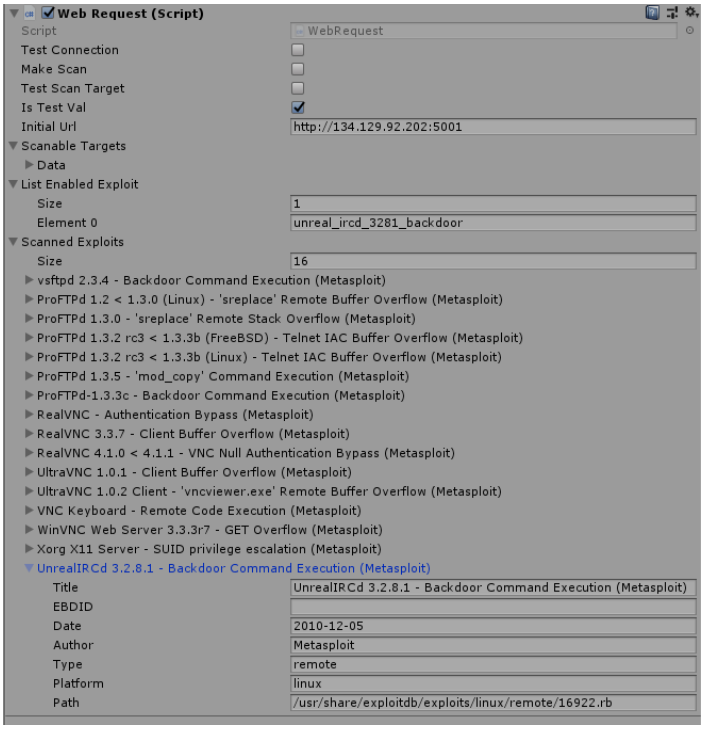


Figure 37. The results of a network scan shown in the Unity Inspector.

The network interface example does not go all the way to automate the hacking process but serves to illustrate how the client-side VR system would communicate with a server with the tools capable of automating an attack.

5.2. Data Visualization Analysis

Some of the first attempts at data visualizations looked at how VR could model data structures in 3D. The first iteration of this was to look at a rectangular building structure for virtual network environments. This shape could show a parent-child relationship between networks and subnetworks, however failed to represent information about interconnected networks, or large intranets which are mainly used by corporations, government, or organizational entities that make up most of the internet.

In the second iteration, the data model mimicked a randomly connected graph to model the interconnected behavior, however the algorithm used to generate the data made little sense since its randomly generated connections made the graph incomprehensible. In order to better understand a network system, the graph would need more information about the types of node and its type of relationship with other nodes. The graph approach started out simple so that it would provide a starting point to create better versions of this model.

The third attempt modeled the data as a tree structure, with the root of each tree being connected to other trees, this would allow for a hierarchy that were like routers providing connectivity to personal computers or server farms. The final visualization allows a more standardized approach, inspired by other works that modeled network data, and the real mapped internet data itself. The final model used in this study would be the basis for developing UI and modeling interaction between the nodes within the graph.

5.3. Visualization Results

The examples for visualization had two very distinct approaches. At first, they were combined into the one primary approach with nodes plotted and stacked on a flat plane, similar to the geographic method used later on. This process left out some information in regards to the hierarchy and relationship to other nodes. The visuals were then updated into a graph format, and this in turn adopted the connections and hierarchy model.

The 2D stacked plots turned into two more different types of geographic models. The first being the regional map where points were plotted across a terrain or surface to be viewed from a bird’s eye view. The second method would show more information such as signal strength from a point of origin. These two geographic methods could later be combined to create both a local and global scale visual representation of nodes.

The result of creating these example graph visualizations is that it showed two primary areas that VR visualization can help in. Which is the connectivity of a network and its geographic locations. These can in future works converge into a global network of connectivity showing geographic plots connected by arcing lines across the surface of the globe.

5.4. Comparison of Motion Data

This section presents the analysis of data gathered to compare different ways the user interacts with the UI. The data collection portion of the project was meant to facilitate evaluation of several of the theories behind UI design choices in VR. The results suggested that several common design beliefs may be invalid and shed light into what advantages or disadvantages each UI design approach had.

5.4.1. Hover-Touch Data

With the hover-touch method, the user hovers their finger over a UI element (for the full report of this, see Figure Appendix: Figure A1, Figure A2 and Figure A3). The UI system figures out which button they are about to hit and highlight that element. This is done by some form of forward vector ray casting that would intersect the element. A combination of forward, backward ray cast, proximity, and velocity estimation of the cursor from the position and rotation of cursor would determine whether an element hit was detected.

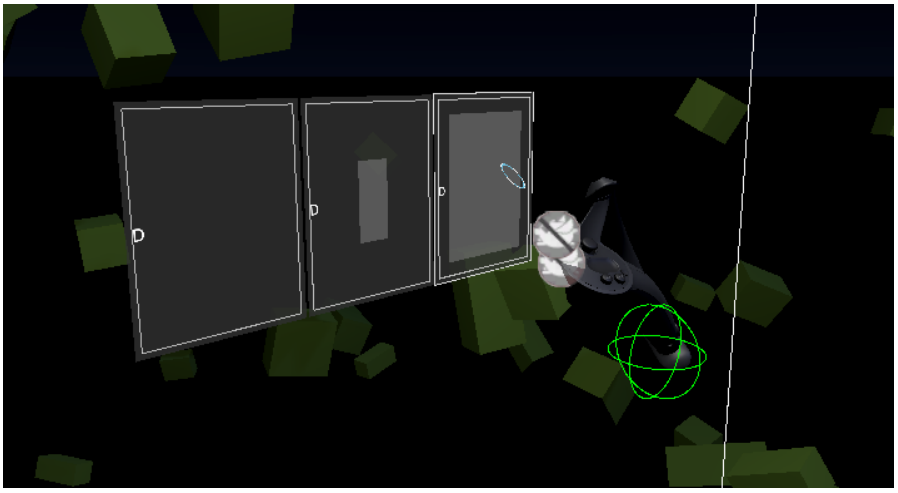


Figure 38. Using Hover UI to record interaction data.

The data in Figure 39 shows the point of origin for the tracked cursor. A destination vector slightly offsets the point using the cursor's facing direction. However, that data cannot be seen with all the movement axes combined since the offset is small and quite insignificant when looking at the data overall (both the origin and target move along the forward vector of the hand, always at the same distance apart). The input value or trigger event is shown as a yellow line representing when the user triggered an in-game element. The Y(up) axis stays mostly the same since the user's hand was level throughout the interaction, the X axis (right) progressed as they moved their hand left to right (note the blue line moving upwards) interacting with each element.

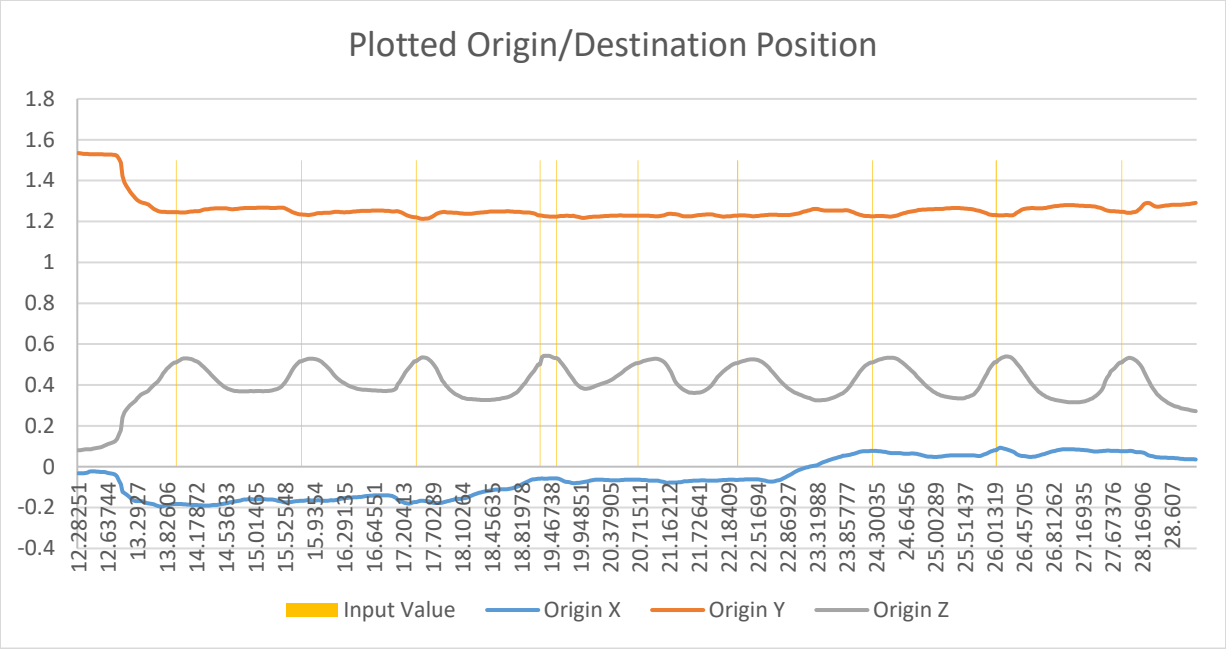


Figure 39. Recorded origin data from the player cursor.

The most interesting feature on this graph is the movement along the Z axis (forward) as the cursor moved into the element firing off the trigger element. Note how the trigger events would occur just before the hand moved back, this indicated a level of consistent reaction from the user that an element had been triggered.

Another interesting feature was the last two events that fired off at the same time, before and after the user reacted. This suggests that the user's cursor was close to another element causing a slight misfire as the user may have moved the cursor to overlap another element to trigger the second event. This behavior accounts for the expected side effect that this method would produce, which is being more prone to error.

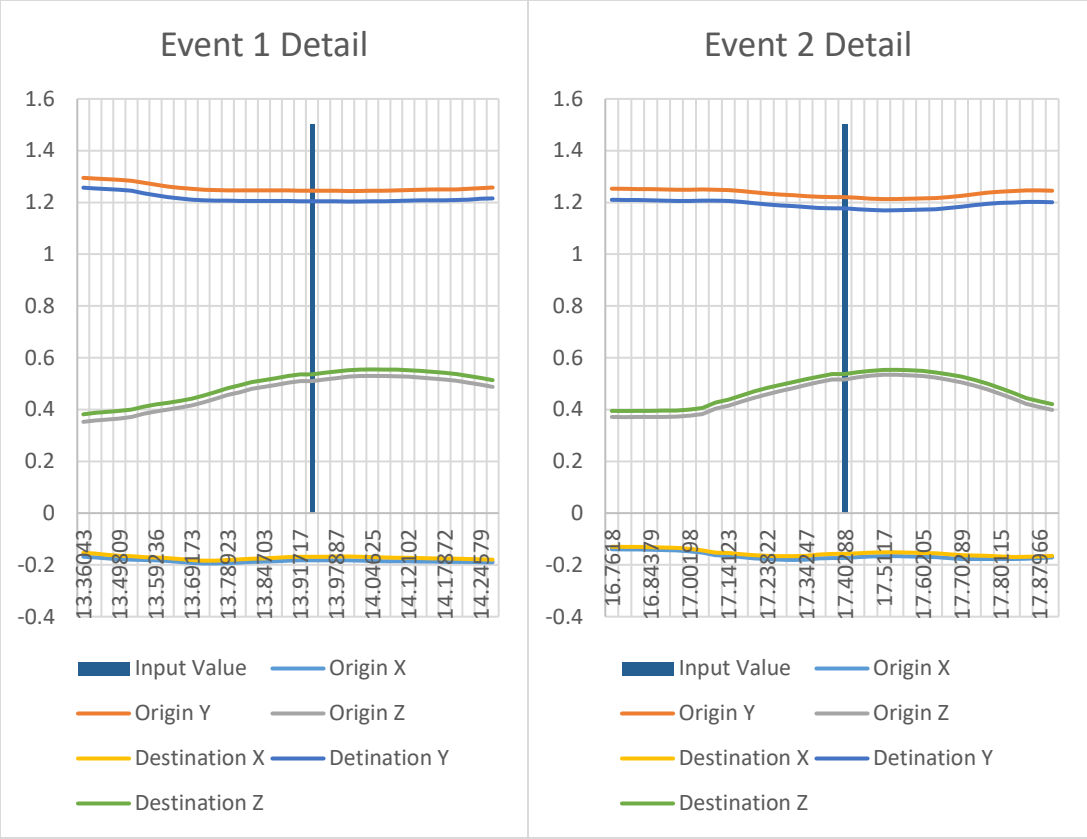


Figure 40. A comparison of the first 2 events triggered in the hover-touch method.

Even though accidental event triggers can happen, this UI method appears to be useful in allowing the user to trigger events fluidly, flowing from one direction to another like they are typing on a keyboard or touch screen. In addition, it also suggests that this UI method may not be good for use with context sensitive events such as execution of a program or careful selection of UI elements.

In a closer look at how each event behaved, the offset of origin and destination values were not significantly distant from each other. The destination is a local forward seeking ray cast point used to obtain the orientation data of the cursor. This is used to simply preserve information for the rotation of the cursor, and thus has little relevance to the conclusions of this experiment. As stated earlier, the event can be seen occurring slightly before the peak of the Z

axis movement, and the player removing their finger out of the element accounts for the Z values going down afterwards.

5.4.2. Pointer Data Results

In this experiment, the user uses a ray-casting pointer to select a UI element (the full results graphed can be found in the graphs appendix Figure A4, Figure A5, and Figure A6). The pointer is a simple visual tool for selecting any object within the game world. In Figure 41, the object that the pointer is selecting are buttons which lies on the flat panels.

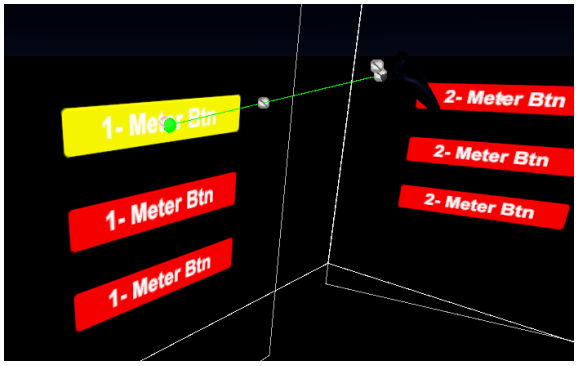


Figure 41. Ray cast from the cursor intersecting with a UI plane.

The setup here are three planes with angles 30 degrees apart and each one, two, and five meters from the user's starting location in the virtual world. The objective of this experiment is to analyze how precision of the destination target changes as distance applies to the UI element selection trigger events. The theory behind this is that, due to the increase in distance the stability of the destination point should decrease. Although this UI does not have good stability at larger virtual distances, it should make sense for interacting with an environment where the elements are farther away, but large enough a target to still easily trigger. Elements such as this could be large buttons on a selection menu or interactable 3D objects represented in the world.

In this series of graphs, Figure 42 and Figure 43, the origin represents the starting point of the pointer, and destination is represented by the ray-casted point on the UI plane. In the graph, a

series of nine trigger events happened, three for each of the different ranges of UI panels. The position of the controller did not need to move much other than rotating to reach each of the buttons.

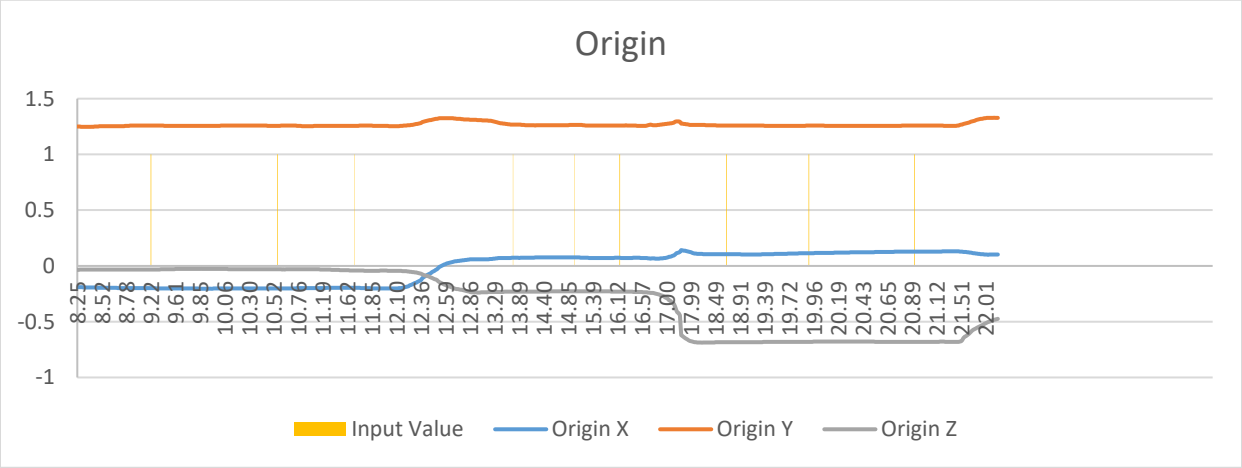


Figure 42. Cursor origin data for the pointer method.

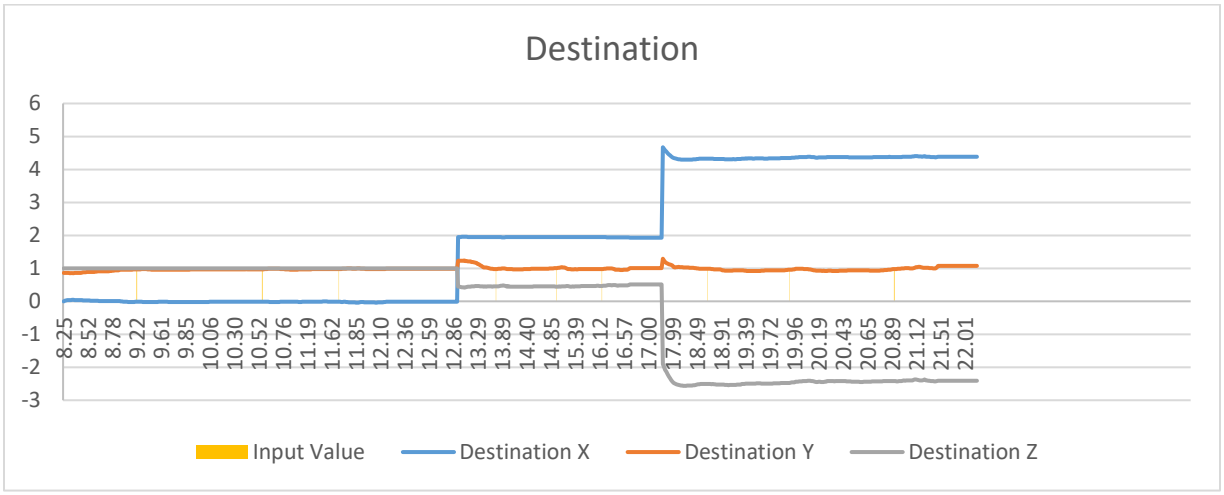


Figure 43. Destination position data, the intersection point of the ray and the plane.

In the destination graph, a subtle change in position can be seen in each button press in the Y axis. For the first set of three events, the UI was flat facing the player and thus the Z axis remained fixed at one. As the player rotate to face the other angled UI planes, the Z axis begins to curve slightly away from the player. The X and Z axis make this correlation as those are the two values that move away from the zero value. This also reveals that not only does distance

play some factor in accuracy but so does angle of the plane. A more perpendicular plane would be easier to interact with, and thus interaction increases as the planes rotate away to become more parallel with the pointer. The difficulty of interaction increases to a maximum if the interaction plane is either parallel or facing away from the pointer. This means that in order to maximize usefulness of 3D planar UI interactions in VR, the designer must attempt to tweak the angle of the plane so that it is mostly perpendicular to the pointer; that is when the user rotates their view each plane should be facing them.

The closer look graphs in Figure 44 show the differences between a ray-cast origin-destination relationship of a one-meter ray-casted trigger event and a five-meter ray-casted trigger event. The instability is hard to notice even in the detailed analysis. In order to look closer at the data, the X and Z axis are eliminated, since they vary based on angle and of the planes. The Y Axis though is relatively invariable and should proportionally change in error rate based on the distance, and thus the following graph was produced to show this.

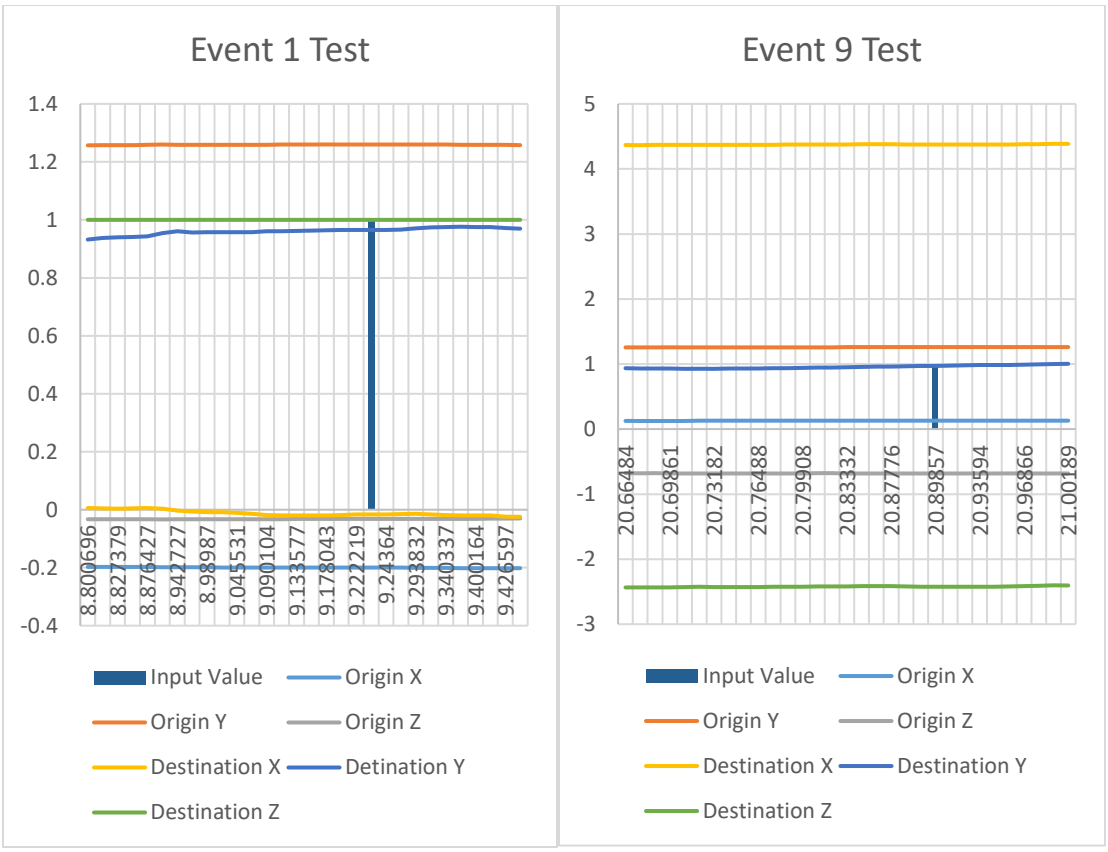


Figure 44. First and last event comparison for the pointer method.

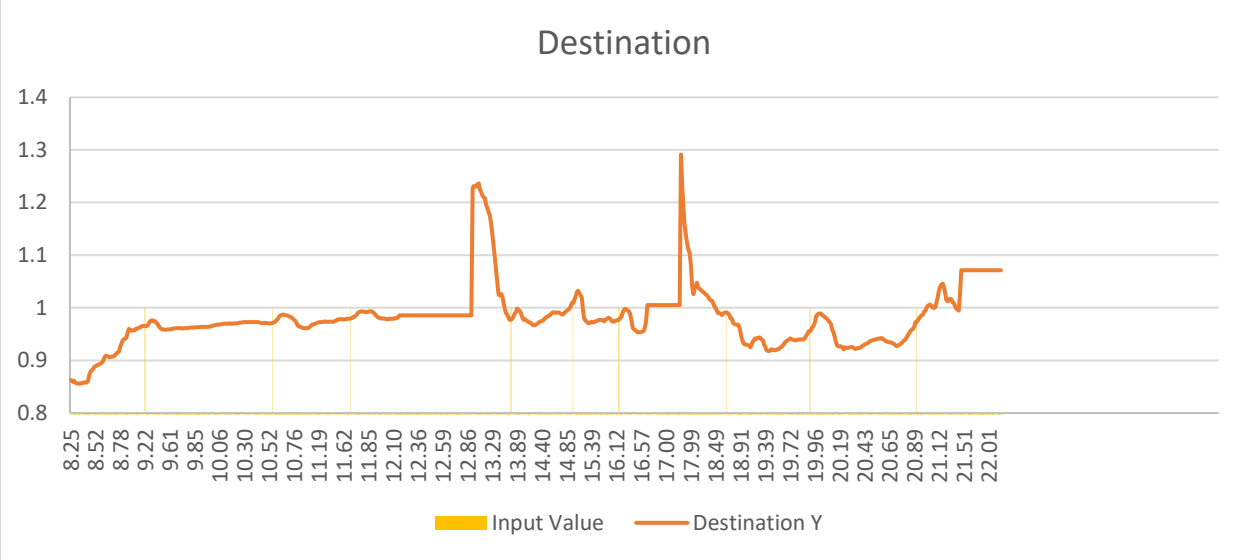


Figure 45. Isolation of the Y axis shows progression of noise through time.

A look at the destination for Y shows that the ray-casted point becomes less stable as distance increases(along the X-Z plane), thus elements in the UI should be scaled to compensate for this. The flat lines in the graph represent missing raycast data as the raycasting stops when the destination point doesn't land on a UI plane. Finally, note the spikes that appear right after each event. These are caused by a trigger press on the controller.

5.4.3. Hover-Click System

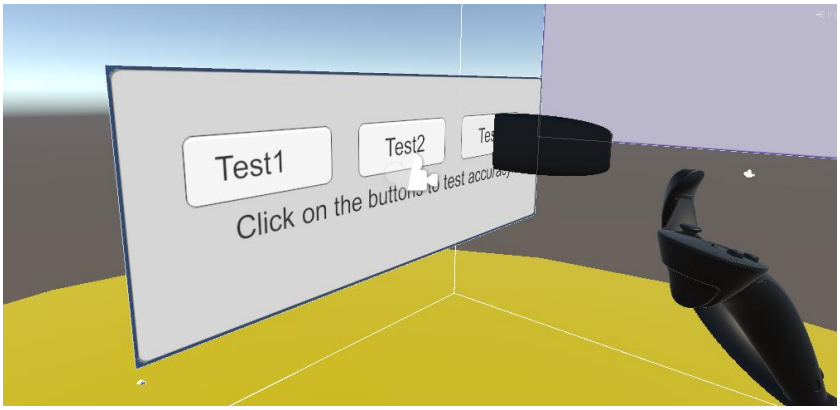


Figure 46. A test environment of the data recording scene for the hover-click method.

In the final test, the hover-click system was used to test out motion in the hand when interacting with the buttons using controller proximity to the element combined with a click to trigger the selected event (the full results graphed can be found in the graphs appendix Figure A7, Figure A8, and Figure A9). This setup consists of three buttons and a cursor that appears only when the player is close enough the button to trigger an event. The hypothesis here is that a button press would interfere with the interaction process which can cause more inaccuracies and slow the number of actions taken per second.

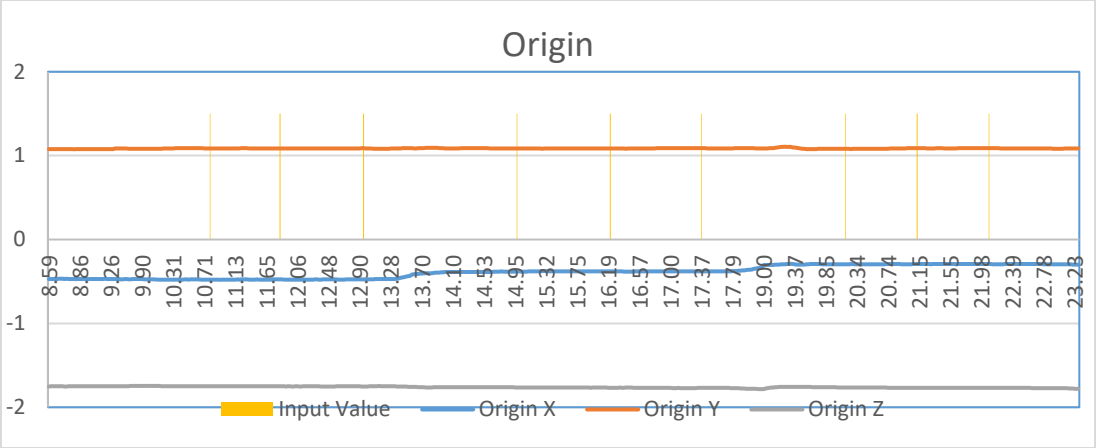
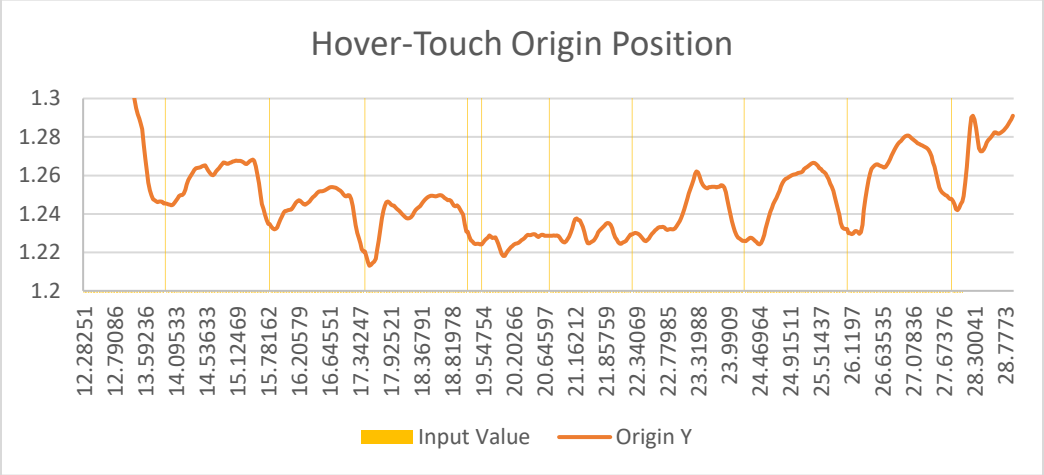
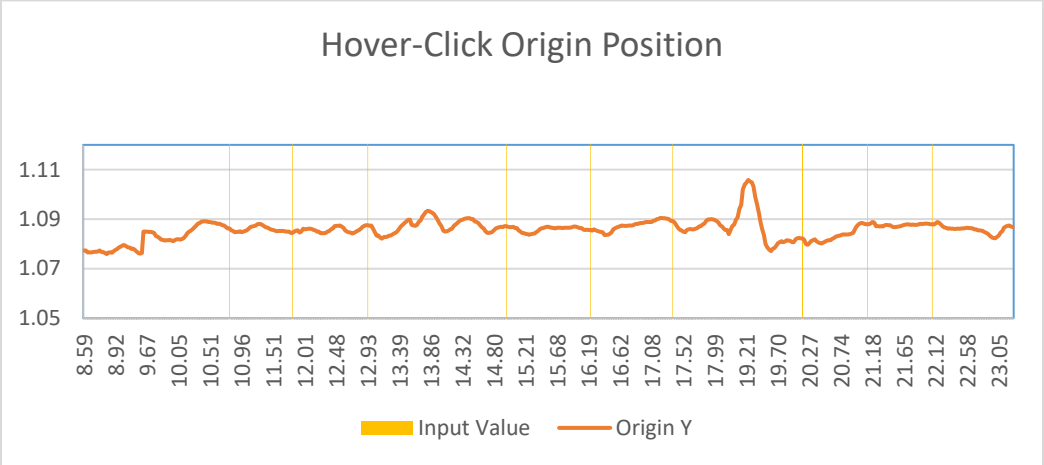


Figure 47. Data representing the origin position of the cursor.

The data recorded between origin and destination would not be very different, since the destination would be the raycase from the UI normal to the player's hand (this method is simialr to the hover-touch method). Thus, the origin and destination are at a fixed distance. A more detailed look at this graph would be to exagarate the Y axis so that the changes can be more noticeable as seen in Figure 48. The next step is to analyze this data to compare its Y axis to the hover-intesect method's Y axis changes.



(a) Movement along Y axis for the origin point of the hover-touch method.



(b) Movement along the Y axis for the origin of the hover-click method.

Figure 48. Comparison of movement along the Y axis in the hover-touch and hover-click methods.

The most surprising result here is that the hover-click method performed better in terms of stability. This meant that the player was less likely to make an error using this method, and that the click event did not affect the accuracy as much as the hover-touch method did. The only advantage the hover-touch method might provide is to be able to use multiple fingers as cursors to leverage multi-touch detection or simulate a keyboard typing behavior. Hover-click succeeds

at providing the best accuracy. Speed of clicking was not a major factor considered in these experiments but is a very good goal for future works building on top of these results.

5.5. Summary of VR-UI Tests

Based on analysis of the data gathered, the following table summarizes each pros and cons of the UI and provides brief explanations of what situations they can be used for.

Table 1. Comparison of VR UI methods.

UI System	Advantages	Disadvantages	General Use Case
Hover-touch	- A quick way to interact with the UI. - Doesn't rely on button presses. - Is useful for building radial hand attached or context driven UI.	- Has less accuracy than its' clicking counterpart. - Easy for user to make mistakes, and detection algorithm is not accurate. - Doesn't work for farther away objects.	- Keyboard UI - When developer does not want to worry about controllers
Pointer	- Great for object interaction that are far away - Only option for UI planes at a distance	- Accuracy is less as objects are farther away. - Relative plane orientation can cause ray to also decrease in accuracy.	- For when objects or UI need to be represented at a distance. - Close up interaction should use a different UI solution.
Hover-click	- Accurate solution for nearby object interaction - A quick way to interact with UI.	- Doesn't work for farther away objects. - Implementation and usefulness rely on the controller or button used.	- Best choice for accurate button interaction for nearby objects. - Does rely on controller but provides better accuracy than free hand interaction.

6. CONCLUSION

The project explored different types of ways to represent network data and UI interaction around the central theme of developing a toolset for a cyber-attack simulation application. The project started with some guesses as to how effective each type of VR based UI pattern would be. After analyzing the data and methods of implementation, the results showed that VR UI systems might be even more versatile than was thought possible. Specifically:

1) Performance of the hover-touch method differed from the hypothesis that it would be the most accurate and simple to use. The test environment and the motion data gathered showed that this method actually had less accuracy due to the heading and speed of movement required to trigger an event. It does also show that it can be used for more keyboard like or fast-moving UI systems such as a UI plane attached to the back of the user's hand.
2) The point-and-click method was inferior at closer ranges, as expected. However, it worked well when elements were farther away. This method works best for elements that were geometric in nature, such as 3D objects placed within the game world. An example would be like using the pointer to select a car out of a parking lot. While this method isn't as useful as a close-up UI system, it makes up for by allowing the player to extend their reach, at the cost of increasing accuracy as distance of interaction increases.
3) The hover-click method was shown to be that it was the best for close range interaction. The user's hands did not pass through, and the cursor moved relatively parallel to the plane of interaction. This mean that accuracy was maximized in this

pattern. The click also did not interfere with accuracy as to the initial hypothesis had believed.

In addition to the knowledge gained from implementing each of the three UI patterns in the application, various other implementations were done to conceptualize the application in VR. These demos include the geographic signal visualizer and the network graph visualizers. The extra dimension in VR allows the user to easily understand the physical location of each node or data point within the system. While these are only visualizations with limited interactive elements, future work can be done to integrate the UI patterns and further control the visuals. The applied use of each of the three design patterns can be seen in the table below.

Table 2. UI interaction patterns.

Cyber-attack Example	Interactions Used			Explanation
	Hover-touch	**Point-and-click**	**Hover-click**	
3D bar graph node visualization.	-	X	-	The pointer system was used for navigating around the bar graphs.
City (local) node visualization.	-	-	X	The hover-click method is mostly used here for navigation and plotting points. The track pad worked best for this since it provided a gradient input system for 2D vectors.
Terrain (regional) node visualization.	-	-	X	The terrain uses the same movement mechanics as the city example. It currently has no other interaction system.
Relational node graph visualization.	-	-	X	The hover-click method was used in implementing a search function. Movement can also be done the same way as the city/terrain examples as well.
Interactive node graph.	X	X	-	The pointer was used as a selection method, whereas the hover-touch method was used for selecting an operation for the node.

From the table of implemented interaction methods, the hover-click and point-and-click methods have the most general uses. The pointer method worked best for an object that is not immediate or close enough for the user to reach out and grab. The hover-click method worked best for navigation. This is due to how the 2D vector input method can proved a gradient between 0-1 for each direction changing the sensitivity of the motion depending on how far away the user presses from the center. An example of this is how a gamepad may have a joystick that can move in any planar direction along with providing a magnitude value. Finally, the hover-touch method has some niche uses for when interacting with keypads. It doesn't provide the

same tactile feedback as the hover-click, but is still useful in some cases where accuracy of input does not matter as much.

6.1. UI Design Methods

A spatially integrated UI design is the most important foundation to VR software development, the ergonomics and intuitive use for each feature rely heavily on where and what the user is interacting with. The initial hypotheses on which UI package would be better would change as the motion data recorded made it clear as to why the hover-touch method would be inferior to the hover-click method in most cases.

The pointer method would work under most situations, but it would become increasingly hard if the ray-cast destination compared to the UI plane's normal increased in angle (0 being a ray casted onto a flat surface head on, and 90 being a ray from the side of the plane). This means that if the ray went from perfectly perpendicular to being parallel to the UI, interaction would become significantly more difficult (and hard to see the elements). This is because ray cast loses accuracy in two configurations, either distance or angle of the ray-cast causes this accuracy to reduce proportionally. However, for non-planar objects, interaction of the object with ray-cast pointers should be the standard. Geometric object interaction accuracy will rely heavily on distance and ray-cast visibility. Objects should take up a significant space in the user's field of view, otherwise the magnified inaccuracy of the pointer will not be helpful for interacting with small or obscured objects in the distance.

It is also worth mentioning that the method that required a confirmation click worked really well since it allowed the user's hand to be the most stable. Hand movement is key to avoiding mistakes in VR interaction. Therefore, if the hand can be tracked along a parallel

surface to the interaction target then triggering the action without moving the hand can yield more accurate interactions.

6.2. Graph Visualization

The project investigated methods for interacting with a data model that represented an interconnected network of computer nodes. The focus in the beginning was to use this model to create a simulated hacking scenario. The result of that was two separate systems; a network enabled automated cyber-attack VR system and an interactive graph visualization.

First, the generated graph node system was meant as a testing system for visualization ideas. The data was used was completely fictional and was generated using a random seed number that allowed for the same model to be generated multiple times. This model was tweaked until a visually comprehensive representation was created.

Second, the network system that simulated hacking utilized a few different tools to model its data. The main tool for creating a network of interconnected nodes was Nmap. However, there wasn't a complex enough real-world available network system (at the time of study) that could be used for the purposes of this research and thus, an example network of two virtual machines on a server was used instead. One of these servers would be the attacking server and the other would the host server. Nmap would return its results in XML of scanned services whereas the attacking server, both built in Kali OS, would return a JSON file exploits found. Both serializers of this data was implemented within the Nighthawk client project. However, because the actual data format was far more complicated than the test data created, they had a similar overall structure, the test data was used to generate different versions of the graph visual. Both the actual data and the generated data represented nodes on a network with ports and the information on what services ran on them. Both models also accounted for the list of possible

exploits on each node. It was clear that formalizing the process would not yield further valuable knowledge on the design of the system, and thus the graph visualization portion was left at the concept and simulation phase.

Finally, the map-like visualization demo focused on a geographic representation of network data using simulated GPS information. This system displayed device signal information in 3D space, overlaid on top of a geographic representation of Manhattan Island. The idea behind going with a purely fictional approach was to create a conceptual visualization, develop concepts for effectively visualizing data, and describe what advantages and disadvantages that approach had. The work on this segment could shed more light for future development of this type of data representation using real world sensors such as drones or mobile phones to track and plot out wireless signals.

The goal of the project was to focus on the design aspects of the UI system to support later work to create an application to hack a network of computers. Visualizing, demonstrating a usable interface, and justifying design choices were the primary outcomes of this research. The final version of the node graph and the signal graph represented viable methods for implementing data visualizations for an automated cyber-attack application.

6.3. Effective Hacking Solutions

To summarize, the overall impression of the Nighthawk VR software does not represent a hacking solution more effective than conventional hacking carried out by trained cyber-security personnel. Although the fully implemented version of this application could be used by high-level authorized personnel to automate a cyber-attack, but they should not fully rely on the automated application for every possible hacking situation. A consultant, or someone knowledgeable in hacking, should still be present when using such an application.

This toolset is designed for users to use these methods in simulating a cyber environment and use it to model real world data such as network traffic flow or as mentioned earlier – geospatial positioning of wireless devices. One of the primary requirements of this application was to be fast and simple to use and avoid error in execution of applications that could unintentionally cripple a network. The area in where the study focused on the most was UI design approaches which seemed to be the most solid portions of the investigation, backed by actual spatial recordings of user interaction with the system. The application could be more complete if some kind of terminal or remote desktop ability was integrated into the VR space. This could become a sort of virtual desktop allowing one person to utilize multiple computers in the hack at once.

6.4. Future Work

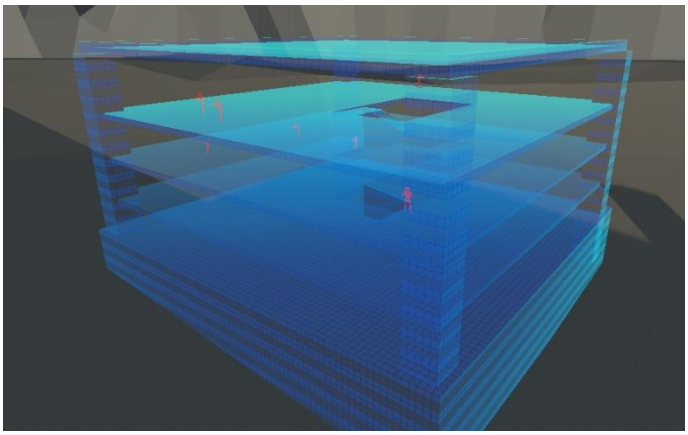


Figure 49. A concept scene of the real-time spatial entity tracking visualizer.

One of the concepts that was looked into was a real time situational based cyber-attack simulation system. This is a system that would build off the geographic system, allowing a VR user to track the location of agents entering a potentially dangerous or hostile area. Their locations and actions, along with the hostile force, could be broadcasted (if the data is available) into the headset's visual representation of the situation.

When storming or raiding a building, police officers or military agents, might be faced with real world danger that could potentially be avoided if the threat was cut off from the outside world during the operation. One of the real-time uses of the system could be to infiltrate a building and monitor the movement of dangerous enemies using sensors gathered from various devices inside the enclosure. This concept could allow devices like phones and web cams to track enemy movement within a building and pinpoint locations of potential hazards. This type of information could be used by experts who can map the layout of a structure and identify safe passage through corridors. The concept would add to the existing knowledge of how network and geographic information can be represented together creating a live cyber view of the world.

REFERENCES

1. "History of Virtual Reality." *Virtual Reality Society,* https://www.vrs.org.uk/virtual-reality/history.html.
2. McFadden, Christopher. "The World's First Commercially Built Flight Simulator: The Link Trainer Blue Box." *Interesting Engineering*, Interesting Engineering, 24 Sept. 2018, https://interestingengineering.com/the-worlds-first-commercially-built-flight-simulator-the-link-trainer-blue-box.
3. "Who Coined the Term 'Virtual Reality'?" *Virtual Reality Society,* https://www.vrs.org.uk/virtual-reality/who-coined-the-term.html.
4. "The Unreleased Sega VR Headset – So Much Effort Squandered." *Virtual Reality Society*, 1 Sept. 2017, https://www.vrs.org.uk/unreleased-sega-vr-headset-much-effort-squandered/.
5. Edwards, Benj. "Unraveling The Enigma Of Nintendo's Virtual Boy, 20 Years Later." *Fast Company*, Fast Company, 2 Feb. 2017, https://www.fastcompany.com/3050016/unraveling-the-enigma-of-nintendos-virtual-boy-20-years-later.
6. Purchese, Robert. "Happy Go Luckey: Meet the 20-Year-Old Creator of Oculus Rift." *Eurogamer.net*, Eurogamer.net, 11 July 2013, https://www.eurogamer.net/articles/2013-07-11-happy-go-luckey-meet-the-20-year-old-creator-of-oculus-rift.
7. Conditt, Jessica. "Valve Is Making a VR Headset and Its Own Steam Machine." *Engadget,* 19 July 2019, https://www.engadget.com/2015/02/23/steamvr-valve-virtual-reality-gdc/.
8. Desnoyers-Stewart, John, et al. "Mixed Reality MIDI Keyboard Demonstration." *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences - AM 17*, 2017, doi:10.1145/3123514.3123560.

9. Chou, Te-Shun, and John Jones. “Developing and Evaluating an Experimental Learning Environment for Cyber Security Education.” *Proceedings of the 19th Annual SIG Conference on Information Technology Education - SIGITE 18*, 2018, doi:10.1145/3241815.3241855.
10. Timchenko, Maxim, and David Starobinski. “A Simple Laboratory Environment for Real-World Offensive Security Education.” *Proceedings of the 46th ACM Technical Symposium on Computer Science Education - SIGCSE 15*, 2015, doi:10.1145/2676723.2677225.
11. Technologies, Unity. “Unity.” *Unity*, https://unity.com/.
12. ValveSoftware. “ValveSoftware/steamvr_unity_plugin.” *GitHub*, 8 Oct. 2019, https://github.com/ValveSoftware/steamvr_unity_plugin.
13. ExtendRealityLtd. “ExtendRealityLtd/VRTK.” *GitHub*, 27 Sept. 2019, https://github.com/ExtendRealityLtd/VRTK.
14. WildStyle69. “WildStyle69/SteamVR_Unity_Toolkit.” *GitHub*, https://github.com/WildStyle69/SteamVR_Unity_Toolkit.
15. Arigo. “Arigo/BaroqueUI.” *GitHub*, https://github.com/arigo/BaroqueUI.
16. Vacheva, Billy. “Review of Star Trek™: Bridge Crew UX & UI.” *Medium*, Virtual Reality UX, 6 June 2018, https://medium.com/vr-ux-ui/review-of-star-trek-bridge-crew-ux-ui-fabc821fa177.
17. Aestheticinteractive. “Aestheticinteractive/Hover-UI-Kit.” *GitHub*, 5 July 2018, https://github.com/aestheticinteractive/Hover-UI-Kit.
18. WRLD3D. “Developer Tools for WRLD Mapping Platform.” *WRLD3D*, WRLD3D, https://www.wrld3d.com/developers.
19. “Mapbox.” *Mapbox*, https://www.mapbox.com/.

20. Burton, Isaac, and Jeremy Straub. “Autonomous Distributed Electronic Warfare System of Systems.” *2019 14th Annual Conference System of Systems Engineering (SoSE)*, 2019, doi:10.1109/sysose.2019.8753838.
21. Tedsluis. “Tedsluis/Nmap.” *GitHub*, https://github.com/tedsluis/nmap.
22. Lyon, Barrett. *The Opte Project*. LyonLabs LLC, 2005, www.opte.org
23. Felton, Michael. “The Virtual Market.” *Opportunities in XR*, Dec. 2017, p. 3.
24. Cipresso, Pietro, et al. “The Past, Present, and Future of Virtual and Augmented Reality Research: A Network and Cluster Analysis of the Literature.” *Frontiers in Psychology*, vol. 9, June 2018, doi:10.3389/fpsyg.2018.02086.
25. Agarwal, Prashant, et al. “Wifi Positioning System Using WLAN Signal Strengths by Block .” *International Journal of Computer Science and Information Technologies*, vol. 5, no. 2, 2014.
26. Barnard, Dom. “History of VR - Timeline of Events and Tech Development.” *VirtualSpeech*, VirtualSpeech, 6 Aug. 2019, https://virtualspeech.com/blog/history-of-vr.
27. Feltham, Jamie, et al. “Surgery Platform Fundamental Increases Authenticity With Eye-Tracking.” *UploadVR*, 16 Aug. 2019, https://uploadvr.com/fundamentalvr-eye-tracking/.
28. Softley, Iain, director. *Hackers*. MGM/UA Distribution Company, 1995.
29. James, Paul. “NASA & Caltech Leaders Aim to Revolutionise Big Data Visualization with VR.” *Road to VR*, 29 Jan. 2017, https://www.roadtovr.com/nasa-caltech-leaders-aim-revolutionise-big-data-visualization-vr/.
30. Nguyen, Tuan C. “You Might Be Surprised to Learn Who Invented Smartphones.” *ThoughtCo*, ThoughtCo, 25 June 2019, https://www.thoughtco.com/history-of-smartphones-4096585.

31. Orland, Kyle. “Valve Data Shows PC VR Ownership Rose Steadily in 2018.” *Ars Technica*, 3 Jan. 2019, https://arstechnica.com/gaming/2019/01/steam-survey-vr-headset-ownership-roughly-doubled-in-2018/.
32. Richter, Felix. “Infographic: The Diverse Potential of VR & AR Applications.” *Statista Infographics*, 6 Apr. 2016, https://www.statista.com/chart/4602/virtual-and-augmented-reality-software-revenue/.
33. Technologies, Unity. “GraphicRaycaster.Raycast.” *Unity*, https://docs.unity3d.com/2018.1/Documentation/ScriptReference/UI.GraphicRaycaster.Raycast.html.
34. “The Lawnmower Man.” 1992.
35. Wolpin, Stewart. “Why The Home VR Market Hasn't Taken Off.” *TWICE*, 23 Jan. 2019, https://www.twice.com/product/why-the-home-vr-market-hasnt-taken-off-ces-2019.
36. “Metasploitable 2.” *Able 2*, https://metasploit.help.rapid7.com/docs/metasploitable-2.

APPENDIX. ALL GRAPHS FROM USER INTERACTION TEST DATA

A.1. Hover-Touch Graphs – X, Y, Z Axis

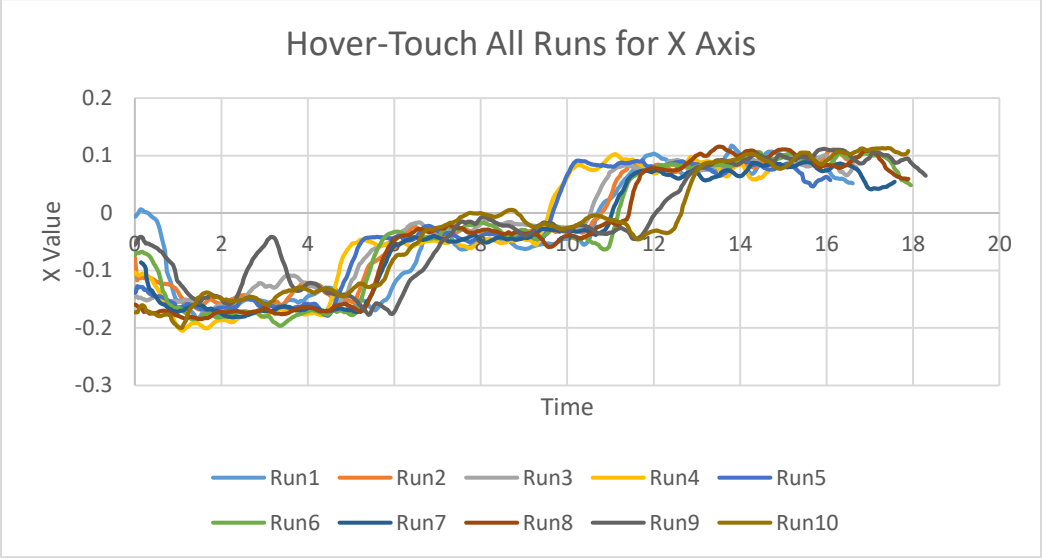


Figure A1. Hover-touch data from 10 sessions for the X axis values only.

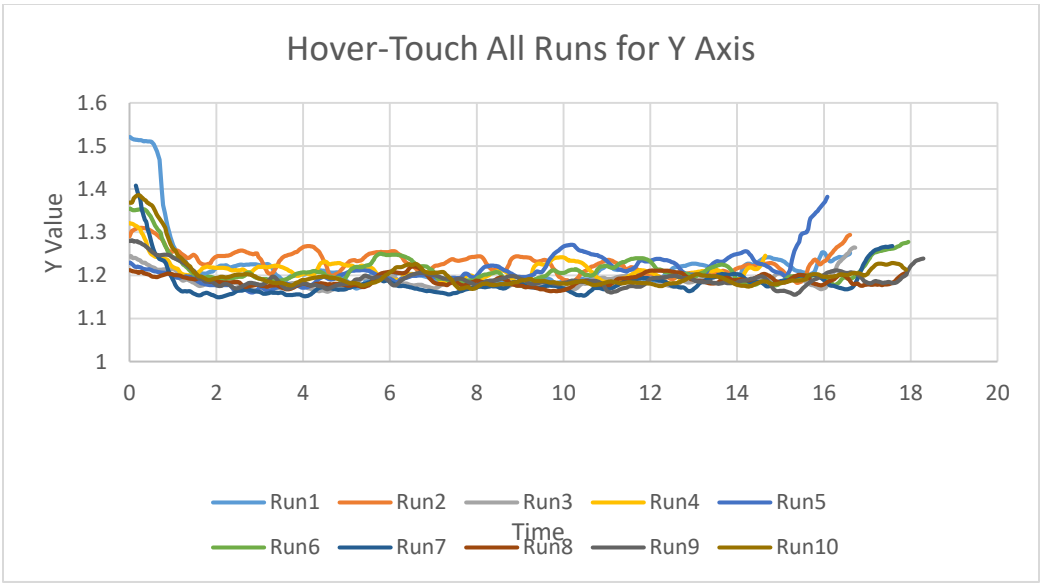


Figure A2. Hover-touch data from 10 sessions for the Y axis values only.

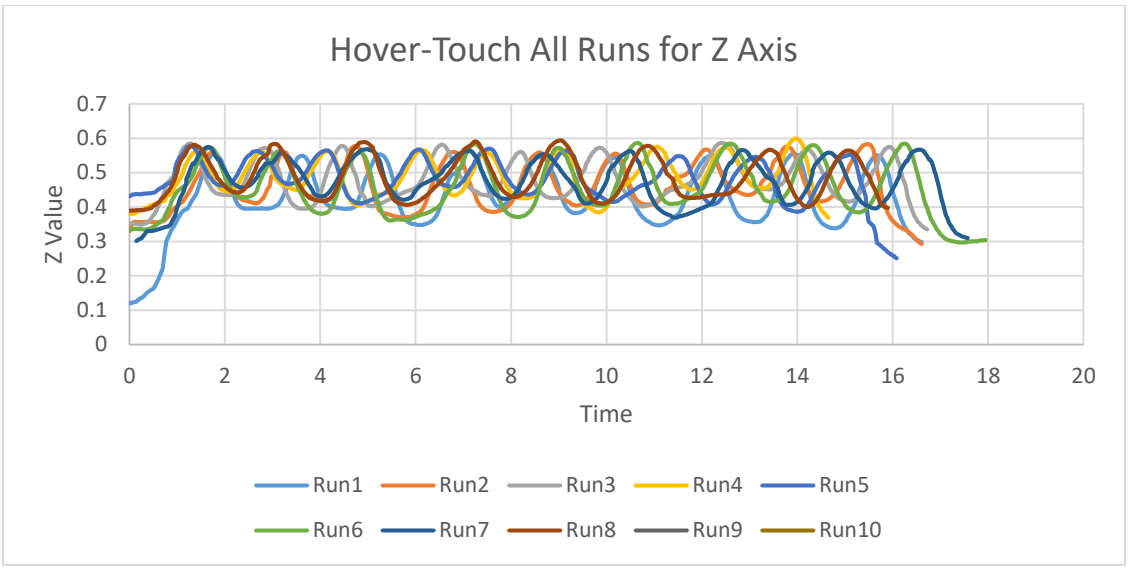


Figure A3. Hover-touch data from 10 sessions for the Z axis values only.

A.2. Point-Click Graphs – X, Y, Z Axis

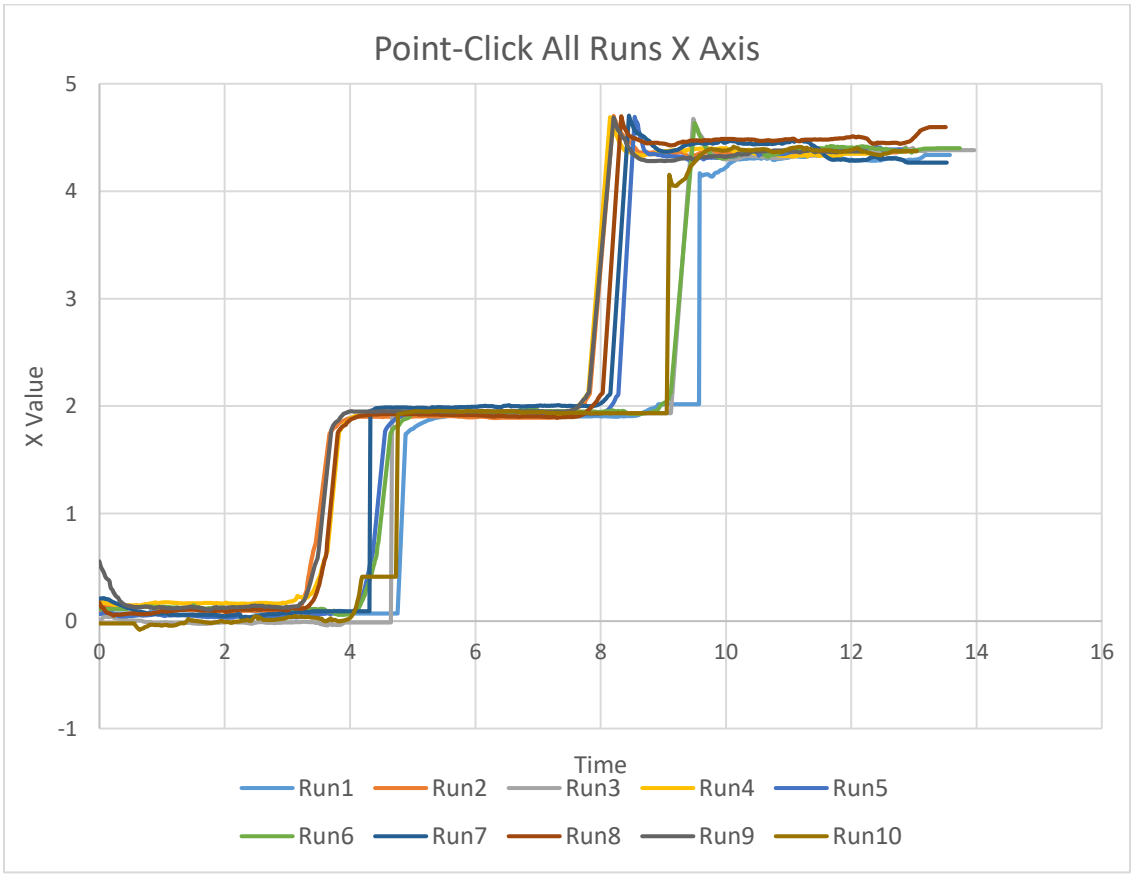


Figure A4. Point-click data from 10 sessions for the X axis values only.

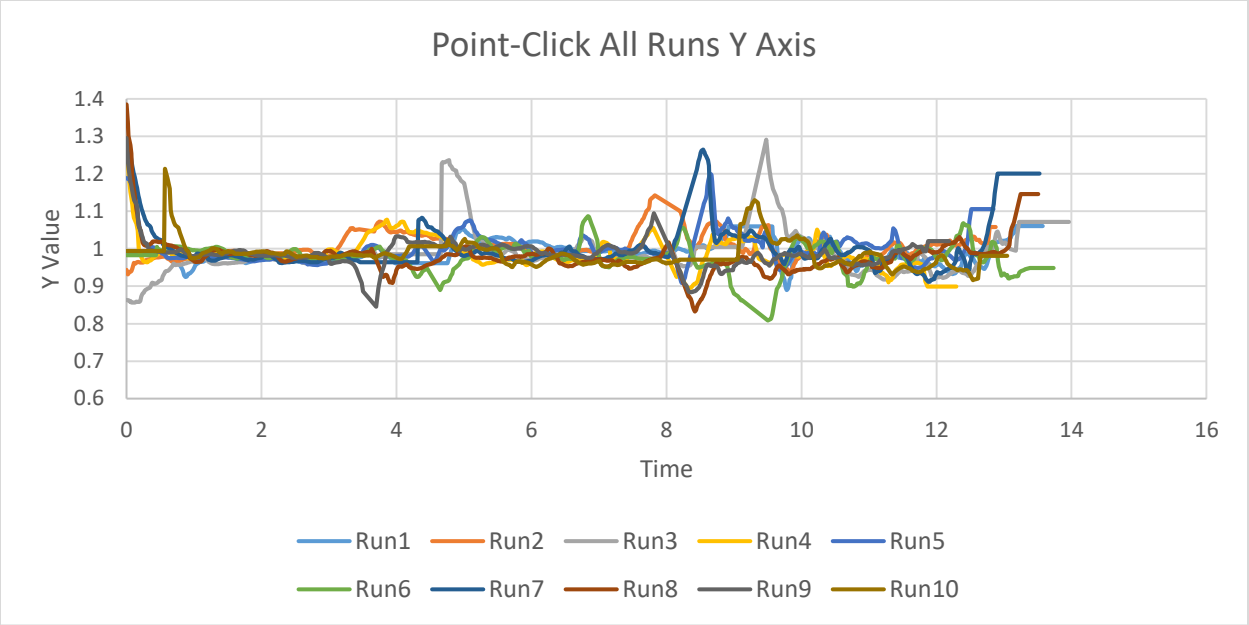


Figure A5. Point-click data from 10 sessions for the Y axis values only.

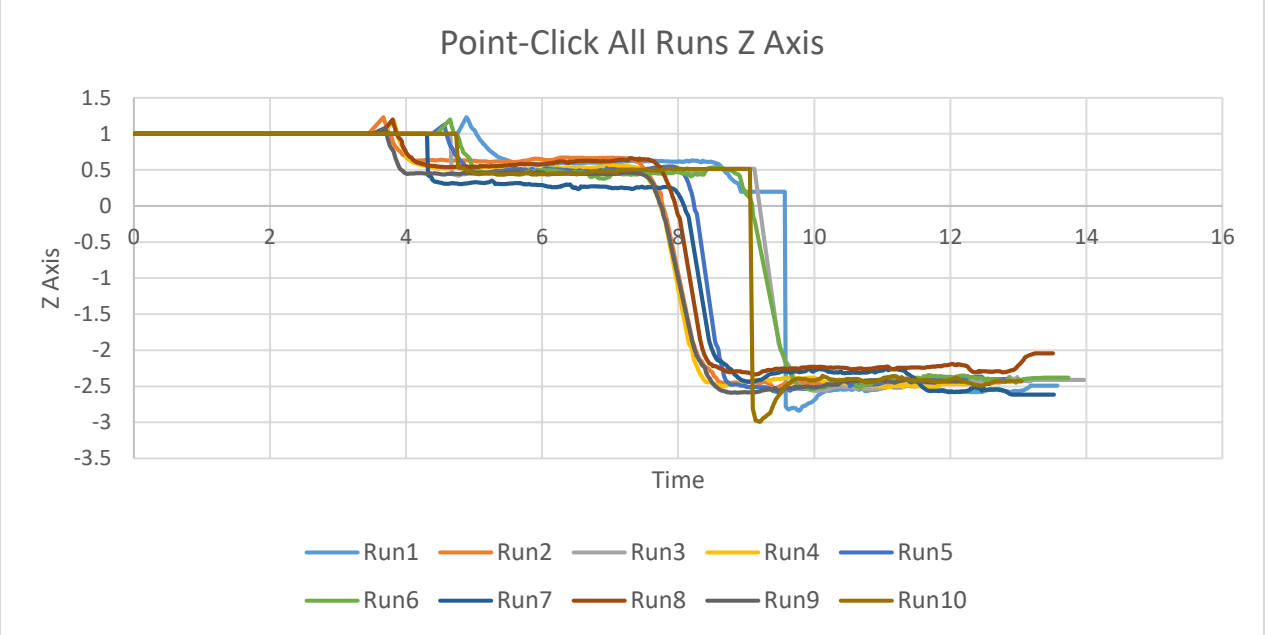


Figure A6. Point-click data from 10 sessions for the Z axis values only.

A.3. Hover-Click Graphs – X, Y, Z Axis

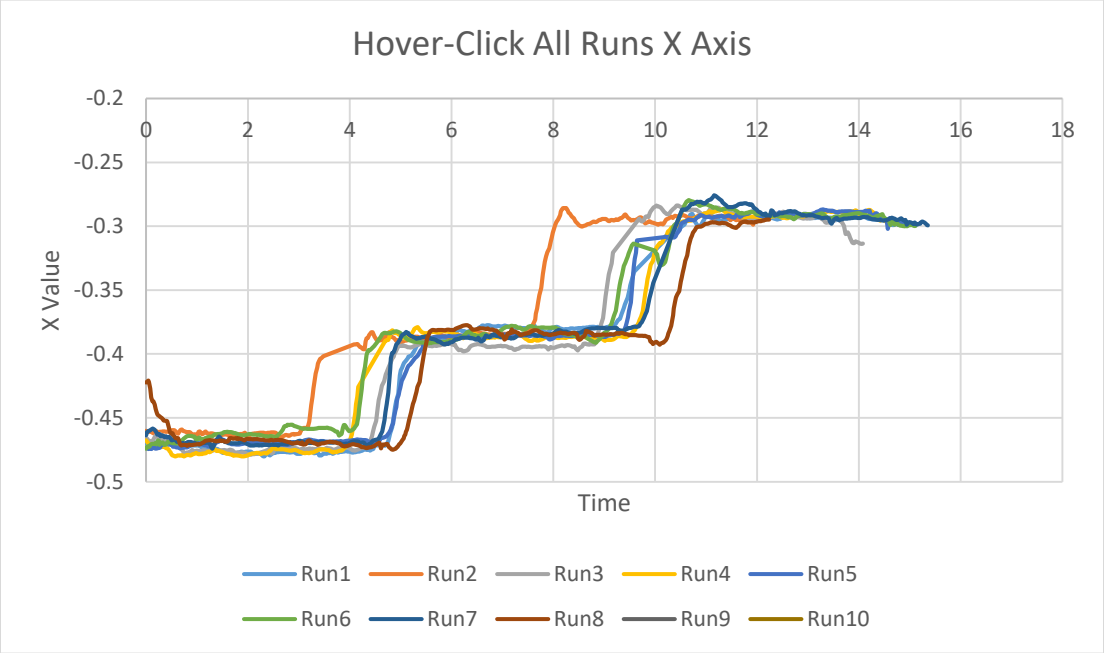


Figure A7. Hover-click data from 10 sessions for the X axis values only.

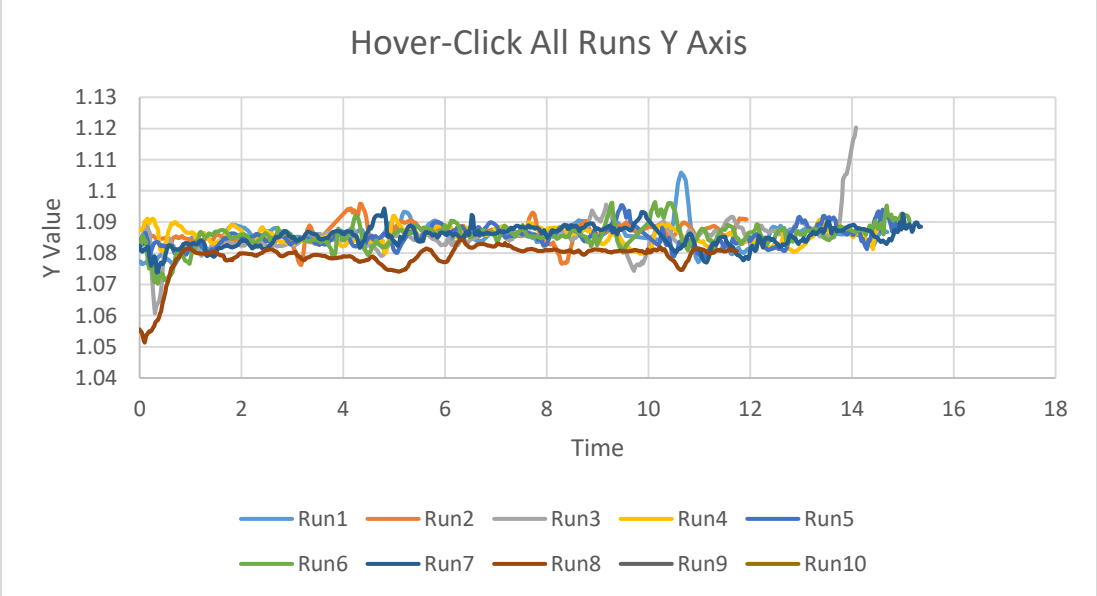


Figure A8. Hover-click data from 10 sessions for the Y axis values only.

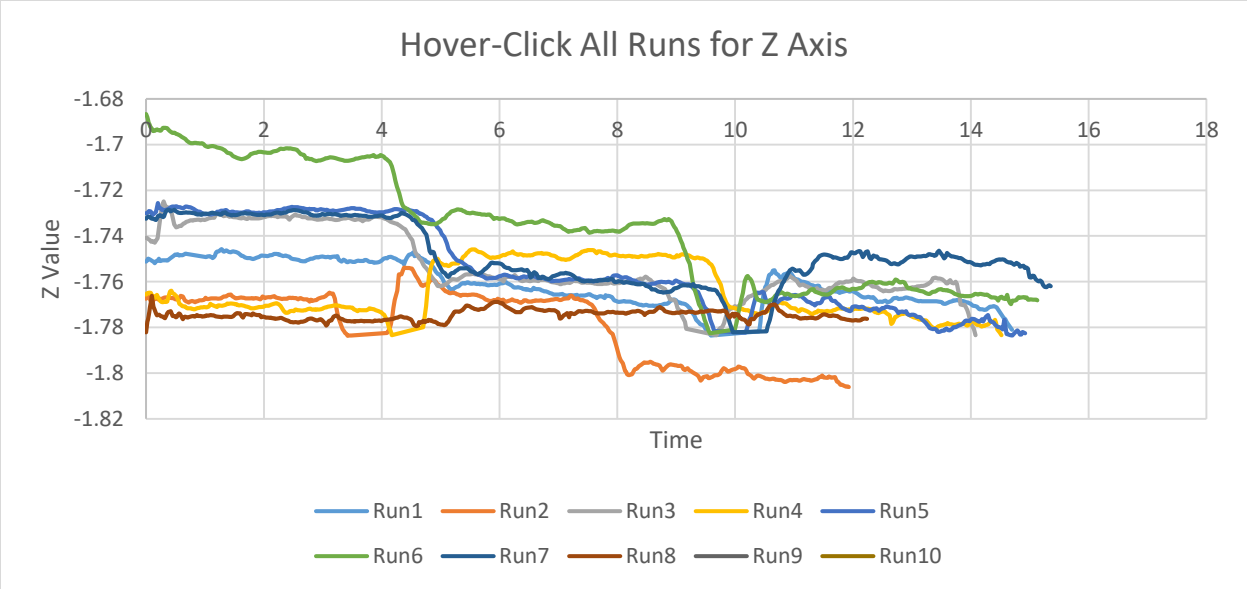


Figure A9. Hover-click data from 10 sessions for the Z axis values only.