Developing User Interface Paradigms for Room-Scale Virtual Reality Systems

Thomas Pritchard
thomas.pritchard1@students.plymouth.ac.uk
Plymouth University

Abstract

We are on the cusp of a paradigm-shift in personal computing. The transition from existing pointer and touch interfaces to virtual and augmented reality, fundamentally changes human-computer interaction. With rapidly advancing virtual reality technology, such as the HTC Vive and Oculus Rift, it is important to develop a considered set of guidelines and best practices to build enjoyable, comfortable, and consistent experiences for users.

This project investigates existing literature surrounding virtual reality interfaces for non-game applications. It experiments with new and existing paradigms for room-scale virtual reality, and produces a set of recommendations aimed at early-adopting developers and platform owners. These recommendations are implemented and demonstrated in a proof-of-concept software application for the HTC Vive.

9740 words.

Table of Contents

Table of Appendices

Table of Figures & Tables

1. Introduction

Virtual Reality Systems (VR) have, in various forms, existed since the 1960s, with products such as the Sensorama (*Heilig, 1962*), and research projects such as the Ivan Edward Sutherland's Sketchpad (*Sutherland, 1963*). Alongside early products and prototypes, fiction, such as *Neuromancer*, *Snow Crash*, and even *Red Dwarf*, delighted in the possibilities, and warned of the potential consequences of ubiquitous VR.

The promise that VR is "*just around the corner*" has been present since the mid-1990s (*Chesher, 1994*), but with the August 2012 announcement of the first Oculus Rift Developer Kit (*Oculus VR, 2012*), high resolution, polished VR experiences seemed both inevitable and imminent. The announcement of the HTC Vive at the Game Developers Conference in March 2015 (*HTC, 2015*) demonstrated the revolutionary possibilities for user experience that room-scale virtual reality systems (RSVR) introduce.

Despite the incredible technological achievements in both the hardware and software of the HTC Vive, a sensible, considered set of user experience paradigms for non-game software has not surfaced. VR interfaces are in their infancy, and so, just as the HCI research of Xerox Parc and Apple was needed to get the WIMP interface of the Apple Lisa and Apple Macintosh, the software development industry needs more research into potential avenues for interacting with VR systems.

The project aimed to develop a clear set of user experience paradigms and guidelines to help designers and developers of RSVR applications provide a comfortable, relatable, and considered experience for their users. To experiment with and refine these guidelines, a data visualisation application was developed as a proof of concept and test case.

2. Background, Objectives, Requirements, & Deliverables

2.1. Background Material

An understanding of the core concepts and products presented in this project report is key to appreciating its results and recommendations. The project requires a basic understanding of the concepts of Human Interface Guidelines (HIGs) (*Appendix 1*), Room-Scale Virtual Reality (RSVR) (*Appendix 2*), and Data Visualisation (*Appendix 3*). More advanced concepts are explained later, but the appendices provide an introduction to these topics.

2.2. Project Objectives

Clear objectives are key to developing an appropriate set of requirements for deliverables (*Van Lamsweerde, 2009*). They also are the measures by which the final products are evaluated. To be fairly evaluated, the project needed to have measurable, realistic objectives.

The overarching goals of the project were to discover, in preliminary research, whether data visualisation can benefit from the technology of VR, and to develop a set of requirements that could be used as the foundation for human interface guidelines.

An overview of the objectives written for the PID can be found in *Appendix 4*, and are evaluated in *§9.2*.

2.3. Project Requirements & Deliverables

The project is comprised of two primary deliverables. The first deliverable is a set of recommendations for interface paradigms that could form the basis for a RSVR platform's human interface guidelines (HIG).

The second deliverable is a piece of software to demonstrate the recommendations in-action. The proof-of-concept software implements the discovered best practices for RSVR. It also provides a regular, non-VR mode for traditional displays to allow for a control group during user testing.

2.3.1. Recommendations for HCI Guidelines

The primary knowledge-creation deliverables are a set of recommendations, found in *§8*, to assist designers and developers developing new human computer interface guidelines for RSVR platforms. The requirements for these recommendations can be found in *Appendix 5*.

The recommendations outline suggestions for RSVR interface paradigms designed to maximise the physical and emotional comfort of users using the system for extended periods. It introduces new paradigms, and develops upon existing paradigms in a way that feels natural in RSVR.

2.3.2. Proof-of-Concept Data Visualisation VR Software

The proof-of-concept data visualisation application demonstrates the recommendations for the HCI guidelines. It is a data-visualisation tool that allows for viewing and selecting geospatial datasets. The included data set is a modified version of the Department of Transportation's 2012 Road Traffic Collision dataset.

A technical explanation of the software deliverable is in *Appendix 6*.

The user requirements (*Appendix 7*) led the design of the proof-of-concept application. These user requirements were told, in accordance with the standard Agile approach, in the form of user stories. The user requirements and the user persona (*Appendix 7*) were used to create the functional requirements (*Appendix 8*).

These requirements were chosen for the core software project deliverable, since it allowed for free experimentation around these requirements. The focus of the project, with regard the software deliverable, was to experiment with interface paradigms to discover what elements worked most effectively in RSVR.

3. Literature Review

In developing new paradigms, and porting established paradigms to a new platform, it is important to study the literature which surround both the new platform, and the existing paradigms.

3.1. User Interfaces in VR

VR interfaces provide a wide range of exciting new opportunities to develop interfaces to enable and delight users. Despite this, existing interface paradigms do not necessarily adapt well from traditional screens. Herndon, van Dam, and Gleicher (*1994*) noted that while some elements may transfer well to VR interfaces, certain WIMP (Windows, Icons, Menus, and Pointers) paradigms, such as menu systems, may not transfer as well. It is possible that, similar to the move away from pointer-based interfaces to touch-screens, a new set of interface paradigms will need to be developed to comfortably and productively use VR applications.

3.1.1. Task-Switching in VR

One of the most apparent issues in transitioning an interface to a virtual environment is moving from a constrained, single-task interface, such as viewing a dataset, to a complex physical interface where the user can be physically interacting with multiple tasks at once. Designing interface elements consistently and simply allow users to recognise paradigms, and allows them to develop complex mental models analogous to interacting with intricate physical environments.

Tanviverdi and Jacob (*2001*) introduced the idea of simplifying design tasks in VR down to their most fundamental atomic units. They discuss how fundamental interactions take place, and how to interpret actions that, while simple to understand in the physical world, might be difficult to model in a virtual world. As an example of this, they give a surgeon in a virtual operating theatre prodding or puncturing a piece of virtual flesh. Both actions are caused by similar movements, but their context reveals their intention. They discussed ways to interpret and prioritise the intended action by mediating between the physical environment of the interaction, and the virtual environmental that the interaction is being acted upon.

Users are able to easily context-shift in VR by moving their head or body. A wide field-of-view enables the user to rapidly context-shift, so it is important to build software interfaces that facilitate this. As input methods change from touch or pointer movement on a two dimensional plane to a motion-tracked controller moving in three dimensional space, the software interfaces need to adapt to accommodate the extended degrees of freedom allocated to the user (*Bowman et al., 2001*).

3.1.2. Movement & Navigation in VR

Movement is a key part of VR, and especially RSVR. Bowman et al. (*2001*) also discussed how, in an immersive virtual environment, it is important to consider a user's movement in two key steps:

- Travel: the process of moving from one area to another area
- Wayfinding: the process of discovering and understanding the route from one area to another.

The HTC Vive's room scale hardware allows for natural, intuitive short-term travel, but presents interesting issues regarding enabling wayfinding and long-distance travel.

Other designers, such as Houldon (*2016*), have investigated movement issues in more recent iterations of VR technology, and provided recommendations for RSVR. The majority of her recommendations focus on accessibility in game-centric software, but many of her findings can also be applied to non-game VR software. One recommendation particularly relevant to a data visualisation application is "*Avoid requiring [the user] to reach down to the ground or high up to the ceiling*". Managing the comfort of the user is key to extended use of the software product (*Vink, 2005*). Reducing required movements to subtle, minute interactions can significantly extend the time a user can continuously and comfortably use a system.

The ease of navigating data sets is an important part of improving the user experience of data visualisation. Researchers at Virginia Tech (*Ni, Bowman and Chen, 2006*) found that a user's ability to navigate virtual environments, and complete tasks reliant on comprehending large amounts of information, improved as the virtual environment displays increased in size. Users using larger (and higher resolution) displays, which filled more of their field-of-view, were able to navigate the virtual environment with less reliance on way-finding aids (maps and a compass). They were able to, on average, complete tasks 62% faster with a higher resolution, larger display. It was also observed that users with larger displays utilised their peripheral vision more, moving their heads instead of just moving the display viewport. Users also reported a better sense of presence with the the larger, higher resolution displays, *"[especially so] when [the users] were seated close to the screen*".

3.1.3. Affordance & Recognition in VR

With new interface technologies, designers often look for metaphors to adapt existing mental models to the new system. The move to touch screens was greeted by wooden panels, rich Corinthian leather, and large, physical-looking buttons. Pressing on a glass screen did not feel like pressing a wooden button, but its appearance exposed its functionality (*Otelsberg, Akshay, and Bhavani, 2013*). These affordances were an important step in the transition from pointer-interfaces to touch-interfaces. A similar level of Affordance will be required for the transition to VR interfaces.

In their SIGCHI workshop, Herndon, van Dam, and Gleicher (*1994*) discussed the options of bringing to three dimensional interfaces interface elements and paradigms from both the real, physical world and from traditional two dimensional interface design. They discovered that when using skeuomorphic, physical metaphors from the real world in VR interface design, screen resolution and input latency were key to making the experience pleasant and relatable.

These issues of hardware specification have largely been solved in the 22 years since the workshop, enabling designers to build out real-world interfaces, borrowing from industrial design and architecture, in virtual environments.

3.2. Data Visualisation

Due to the sparse amount of literature available specifically for designing data visualisation software for modern VR environments, it is important to draw upon bodies of work that, while not specific to VR, are still applicable to the software project.

3.2.1. Geovisualisation

The representation of geospatial data on map-like interfaces is known as geovisualisation (*MacEachren et al., 2004*). One of the core ideas of geovisualisation is that spatial environments can provide a better environment for representing data, especially data that has space and time vectors (*Geovisualisation, 2011*).

Geovisualisation has helped scientists for over a hundred years (although the term has only been used since 1953) discover and document correlations and trends between locations and geospatial data. In 1854, John Snow, a physician, discovered the link between cholera and the locations of water sources, which resulted in the discovery that cholera was waterborne, and not airborne as was previously suspected (*Snow, 1855*). The clusters of cholera were centered around water sources (primarily the water pump on Broad Street), leading Snow to hypothesize that there could be a causal relationship between the two. Geovisualisation enabled John Snow to discover correlations that would have been very difficult to observe with raw data.

Geospatial data can become complex very quickly. Large amounts of data set out over a large area, especially if there are multiple dimensions of data, can be difficult and confusing to interpret and understand. The ability to comprehend and make fast and accurate decisions based on geospatial data correlates to the amount of spatial orientation the user has over the data set (*Swink and Speier, 1999*). Swink and Speier also found that aggregation and dispersion of data had effects on how well users were able to make use of the information presented.

3.2.2. Designing Data Visualisations

Well-designed data visualisations can inform and educate users, whereas poorly designed data can confuse and annoy them. The layout of data on a page, or a screen, can help its readers to understand and find new insights from the dataset. The absolute aim of data visualisation is insight (*Van Wijk, 2005*).

The presentation of information related to a data point can help investigation and exploration of a data set. The way this information is presented, however, can reveal the biases of the designer, and so careful care should be taken to present the data as sourced (*UC San Diego, n.d.*).

Edward Tufte, possibly the most well-respected designer of data visualisations, uses weight of type to signify importance, laying information out in grids (*Tufte and Graves-Morris, 1983*). Tufte discussed how the spatial layout of the information can make it easier for readers to compare two similar data points. Once the labels have been read to understand what the data represents, the reader can refer to that data's position on the page, or in the panel, to compare between similar data points.

4. Legal, Social, Ethical, & Professional Issues

4.1. Data & Open Source Licensing

With open source code and data, it is our legal and moral responsibility to abide, to the best of our knowledge, by the licences that the information is licensed under (*Krogh et al., 2012*). With the technology in its infancy, a vibrant open source community is important to spreading RSVR experiences with those designing, developing, and using the hardware. In order to foster this community, it is important to comply with existing terms of open source software, and return knowledge to the community with an accessible license.

4.1.1. Compliance with Data License

The dataset modified for use in this software application is the "Road Safety Data" dataset (*Department of Transport, 2015*), published by the Department for Transport using the Open Government License Version 3 (*UK Government: National Archives, 2016*). Evaluation of the Open Government License can be found in *Appendix 9*.

4.1.2. Compliance with included Open Source projects

The software project includes information from two open source projects:

- OpenVR (*Valve Software, 2016*), a runtime developed by Valve Software to allow developers to develop for multiple VR hardware vendors without specific knowledge of the hardware they are targeting.
- RTS Camera (*Sylkin, 2016*), a Unity plugin developed by Denis Sylkin to facilitate quick and easy development RTS-esque games.

Compliance with these project's licenses is outlined in *Appendix 10*.

4.1.3. Open Source License for Project

From the outset, the project aimed to be open and accessible, sharing knowledge, recommendations, and code for the benefit of the VR community. The software application was open sourced and available to designers and developers of RSVR software. The project is licensed under the MIT Open Source License (*Opensource.org, 2016*).

4.2. Data Representation

It is the social responsibility of the those involved with the project that the software proof-of-concept represents the datasets fairly and without bias. The goal of this data visualisation software was to show the data as it is found, with a politically neutral perspective. The data was modified slightly with these issues in mind, to ease presentation, as explained in *Appendix 11*.

4.3. Opportunities in Visualising Open Data

The decision was made early on to open source the proof-of-concept as well as the HIG recommendations. Open source software can enable software designers and developers to build data visualisation tools, which can enable the next generation of social researchers.

Giving data scientists, sociologists, and other researchers better tools to visualise and comprehend large datasets provides them the opportunities to better discover new trends in open data that can improve the standard of living for everyday people. Free, open, and effective data visualisation tools can enable those who might not otherwise have had the means by which to discover information to help their communities.

4.4. Health & Safety Issues

With new interface technology, an important part of developing paradigms for continued, productive use is understanding the health implications of the new technology, and building software that works with the hardware to provide a comfortable, safe environment for getting work done.

4.4.1. Physical Safety in Tethered VR Systems

RSVR requires the user to cover their eyesight entirely with a HMD, causing a total blackout of their physical environment, and replacing it with a virtual environment. Because the user is moving around in the physical environment, but only has the visual stimulus of the virtual environment, it is important that the physical environment is set up in a way such that it is safe for the user to move around in. An evaluation of the requirements for safety in physical environments was undertaken (*Appendix 12*).

4.4.2. Physical & Emotional Comfort in Virtual Environments

The physical wellbeing of the user in the virtual environment is as important as their safety in their physical environment. Though considered less often in literature, emotional comfort in virtual environments is also incredibly important. To ensure the user is relaxed, comfortable, and prepared for working in VR for extended periods, their comfort in virtual environments must be considered. Analysis of the requirements to maximise the comfort of users in virtual environments was undertaken (*Appendix 13*).

5. Methods of Approach

5.1 Software Development Workflow

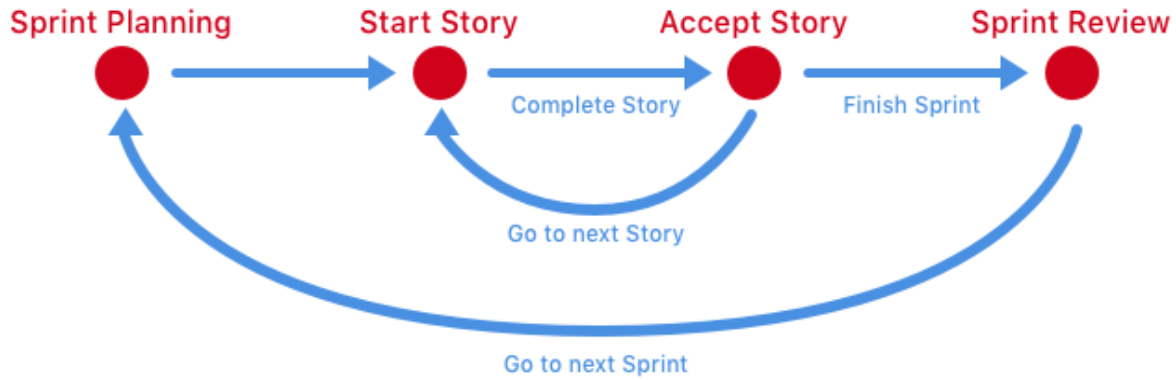

***Figure 1**: A diagram of the Agile software development workflow.*

The project was developed using the 'Agile' software development workflow (*Martin, 2003*). Agile was chosen as it enables fast feedback loops (*Figure 1*): building features, testing them with users, adjusting it in response to user feedback, and continuing to test until the feature is polished. The feedback cycle of just two weeks (one sprint), means if an avenue of discovery turns out to be a dead end, it can be discovered early and the sprint plan can be altered to explore other designs. When an experiment, such as the self-scaling experiment, isn't going well, it is quick and easy, from a project management perspective, to pivot and move forward in another direction.

Sprint reviews and sprint planning are vital parts of the Agile project management workflow (*Schwaber, 1997*). These meetings are used to evaluate how a sprint went, and prepare a set of work for the next sprint. These are typically scheduled following a sprint, and before the beginning of the next sprint and attended by all members of the design and development teams, project managers, and owners. The agile workflow supported the project reporting requirements and provided a framework for review and planning meetings with the project supervisor.

5.2. Project Management

The Agile workflow employed during the project was a straightforward Scrum methodology (*Schwaber, 1997*). Scrum's daily standup meetings, which are usually between the team, but in this project were only attended by the sole author, were personal reviews of the current sprint's planned stories and a review of the day's tasks and events. It also enabled the author to view the current progress in light of the entire project's direction.

Sprint reviews and planning took place during alternate weekly meetings between the author and the supervisor. The weekly meetings between the author and the supervisor allowed for a mini-reflection mid-way through sprints, allowing for course correction in the event that the

author could not see the forest for the trees. In the event that an in-person meeting was not possible, email updates were be provided, and an alternative time was arranged to briefly discuss any issues that arose. Members of the internal studio were also available to discuss issues if the supervisor was not available. After Easter, the review meetings took place on an 'as-needed' basis.

Risk analysis also took place (*Appendix 14*). These issues are discussed in more length in the End of Project Report (*§9)* and the Post-Mortem (*§10)*.

5.3. Source Code Version Control

The software project and its source code were managed using the Git version control system (*Git, n.d.*), as well as GitHub as a remote Git server. The project used the standard feature branching workflow for each experiment, with changes being merged back into the master branch. Since the project was performed alone, pull requests were not used when merging into protected branches (in this case, the master branch).

6. Architecture

The project required matching both hardware and software to create a comfortable, polished solution. As a result, the architecture of both hardware and software were vitally important to the success of the project.

6.1. Hardware

Figure 2*: The HTC Vive's final release version. Hardware similar to this was used throughout development to build the software system. The HMD is seen (centre), with the tracked controllers (bottom corners), and the lighthouse infrared systems (top corners).*

The project used the HTC Vive RSVR system (*Figure 2*), comprised of a position-tracked HMD (Head Mounted Display), two position-tracked handheld controllers, and two tripod-mounted "Lighthouses", boxes which flood the room with wavelengths of light invisible to the human eye that are used by the tracked objects (the HMD and the controllers) to track their position and rotation in the room.

6.1.1. Using the HTC Vive

The HTC Vive RSVR system was used during the development of the project. At the beginning of the project the first-generation developer kit was used, and upon delivery of the HTC Vive Pre developer kit in March, development and testing was transitioned to the new developer kit. The HTC Vive was chosen primarily as the first, and best established RSVR system available. It allowed for sub-millimetre tracking in a room up to $4m^2$, which allowed experimentation with large movement and gesture-based interfaces. Requirements for physical space can be read in *Appendix 17*.

The recommendations outlined in the deliverables are not specific to the HTC Vive hardware, and can be generalised to any similar RSVR system. Most of the recommendations can also be applied to general sitting-or-standing-scale VR systems (such as the Oculus Rift or

PlayStation VR). To use “VR Mode” in the proof-of-concept, however, a HTC Vive is required.

6.1.2. Using In-Development Hardware, Firmware, & Software

For all but the final month of the project, the HTC Vive was not publicly available for purchase. The development kits were procured through established personal development relationships with Valve Software and HTC. This early access to the cutting-edge technology added a layer of complexity to the project, introducing a number of difficulties that needed to be addressed or mitigated.

The pre-release software development kits were largely undocumented, which required careful analysis of the source code files to determine API functionality. A lot of early development was completed without any official OpenVR documentation, greatly increasing the difficulty of implementing new paradigms in new technology. Analysis and the results of developing using very early stage hardware are discussed in depth in the Post-Mortem (*§10.2*).

6.2. Programming Language & Development Environment

The software stack was just as important as the hardware stack; a good stack would enable rapid prototyping and iteration, leading to a better, more refined software product.

The Unity game engine was used throughout the project, in both “VR Mode” and the “Non-VR Mode” of the software solution. An in-depth discussion of the reasons for using Unity during development, including how it was integrated into the VR workflow, is included in *Appendix 15*.

The C# programming language was used, along with the Visual Studio IDE, for the development of the scripts and modules that enable functionality in Unity. A discussion as to why C# was chosen for development can be found in *Appendix 16*.

7. User Experience Experiments

The project was built on three main experimental stages to discover and refine interface concepts and paradigms that transfer well to RSVR. The early stage of the hardware, the underdeveloped existing interface paradigms, and the experimental nature of the project meant there was not a clear idea of how the final deliverable would look or work at the beginning of the project.

The experiments allowed for rapid iteration on ideas and interface elements, and meant elements that were not working well could be discarded or folded into alternative designs. Each experiment lasted between three and four weeks, allowing for development and a small amount of informal testing.

7.1. Experiment #1 (Scale User)

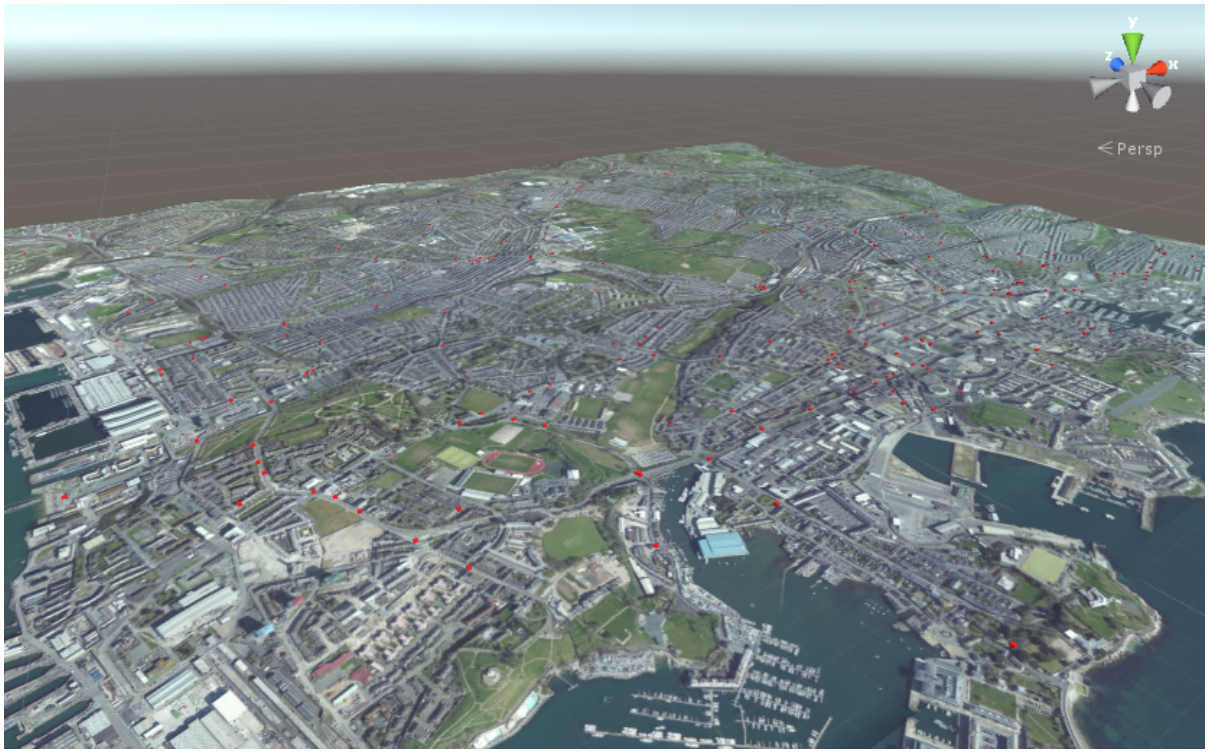

***Figure 3**: The terrain with data points on it in Experiment #1.*

Getting maps, data, and basic interactions into a virtual environment were the key goals of the first experiment (*Figure 3*). This experiment needed a lot of upfront development to build the systems required to facilitate the later experiments.

One area that was explored in the first experiment was scaling and positioning the user. The user needed to have control over their size and position to be able to see the data at both a macro-level, to get an overall idea of the trends and correlations, and the micro-level, to get more information about individual areas or data points.

Another idea explored was a large display ~10m from the user. The display had the selected point's data displayed on it, and would rotate around the central point of the environment, always being in front of the user if they look up at it.

7.1.1. Technical Implementation

Map information was created using the standard Unity terrain system, along with height maps and diffuse maps baked from the Bing Maps API of the Plymouth area. The map for the proof-of-concept was baked offline, but for a real-time product for geospatial data visualisation this could be integrated to coincide with the user is loading the dataset.

The modified dataset (*Appendix 11*), a set of road traffic collisions (RTCs), was loaded from a CSV file, containing location information and data, such as time and accident severity. Significant difficulty occurred in converting latitude and longitude coordinates to Unity's Cartesian Coordinates (*Appendix 18*).

7.1.2. Data Selection

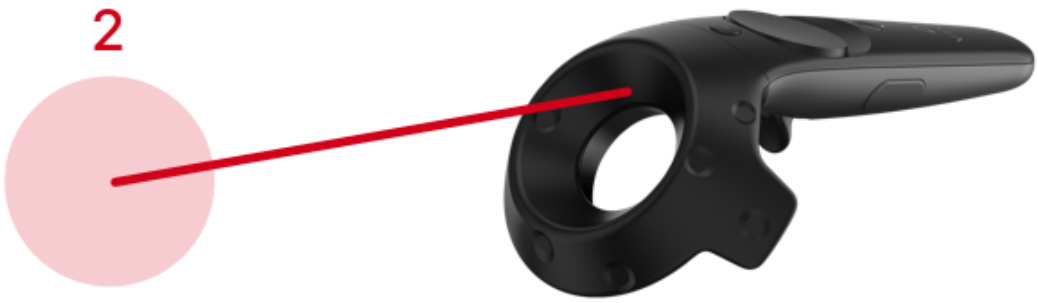

***Figure 4**: A diagram of the controller system to select data points in Experiment #1. A modified version also appeared in Experiment #2.*

A system was required to select individual data points. The device needed to work with data ranging from floor-height to waist height, be able to select data that could be further away from the user than they would like to move, and be able to select individual data points in dense parts of data.

The solution was a laser-pointer originating from the left controller (*Figure 4*), with a sphere at the end point of the pointer. The sphere had a number above it, indicating the number of selectable points. The laser pointer could be pointed at the map, tracing over it. The user could select a point by pulling the trigger on the controller. If more than one data point was available to select, pulling the trigger subsequent times would cycle through the available data points.

7.1.3. Conclusions & Findings

After informal testing with a small number of users, it was reported that users found that RSVR maps allowed them to have a better sense of space, position, and scale than with traditional displays. They did, however, not like managing their own scale, and found the location of the pointer to be unreliable. For an extended look at the conclusions drawn, please see *Appendix 19*.

7.2. Experiment #2 (Table Map)

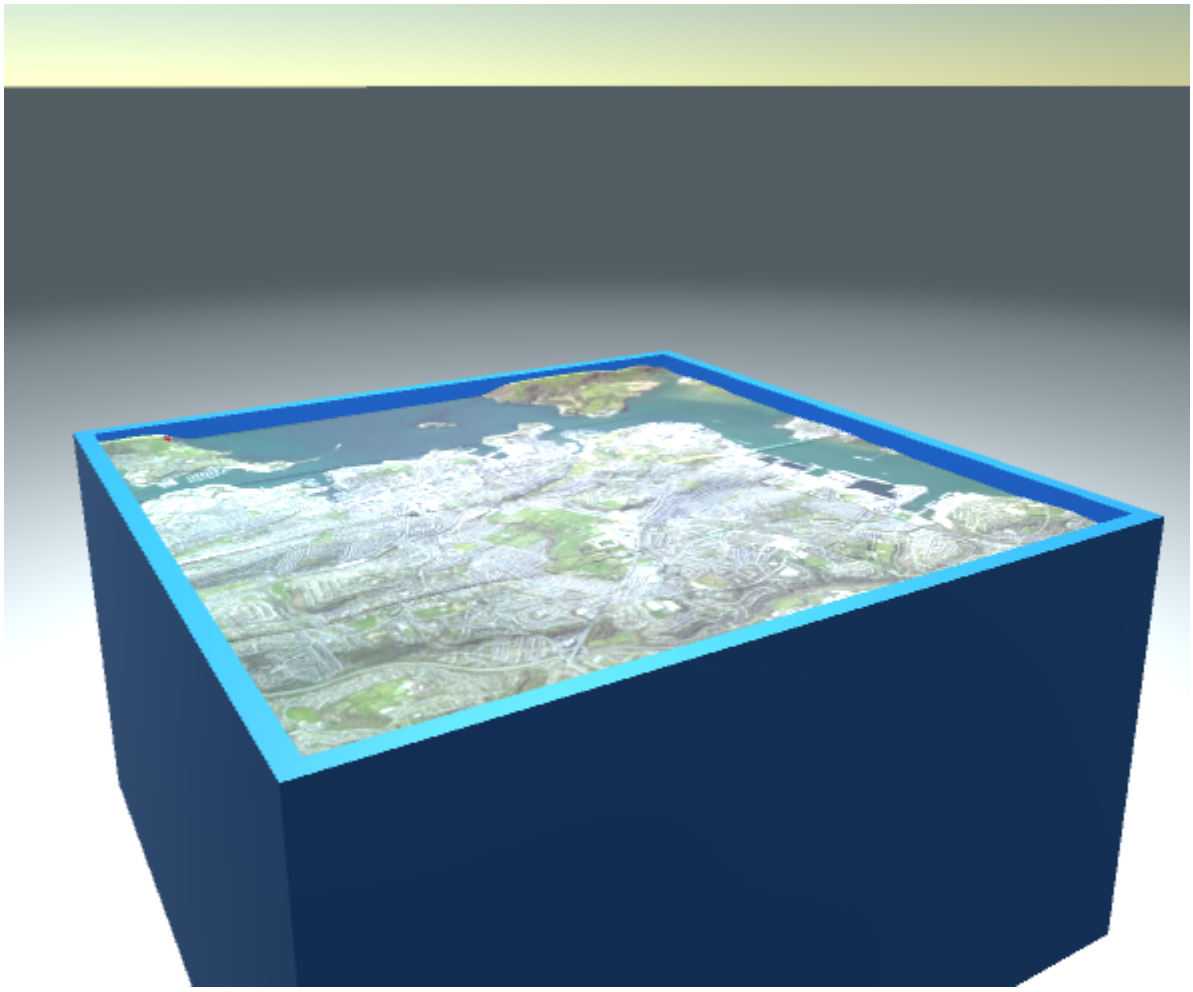

***Figure 5**: The table-interface used in Experiment #2. A user can walk around the map, which is set in a 1.5m^2 table in the center of the room.*

Three problems identified in the conclusion of Experiment #1 needed to be solved:

- There should be less need for managing personal scale and position in a way that requires a conscious cognitive load.
- The laser-pointer should be more predictable
- Data information should be more visible next to the data points on the map.

7.2.1. Table Interfaces

This experiment focused on a table-based interface (*Figure 5*). Here, the user was not required to manage their own scale, and managed their position by moving around the table. If the user wanted to see more information, they could simply move closer to the data.

Alongside the table interface, the experiment also altered the laser pointers to have a fixed distance, regardless of their interaction with the terrain on the table. This meant that any movement in the hand resulted in a one-to-one change in the end of the pointer.

While out-of-scope for this project, a table interface would suit a multi-user networked virtual environment, facilitating collaboration amongst a number of colleagues potentially hundreds or thousands of miles apart.

7.2.2. Data Panels on Virtual Controllers

To solve the issue of having the data information distant from the data points on the map, a solution was devised to have a slightly simplified data panel containing the data information to the left of the pointer controller. This allowed for viewing the information in the same field-of-view as the data point, ensuring the user could see the information about the data point in context. It was necessary to develop a new shader (*Appendix 20*) for the data presentation.

7.2.3. Conclusions & Findings

At the end of the second experiment, as with the first, informal testing took place. Users reported that it was easy for them to understand the dataset and its spatial relationship with the map.

Users benefited from not needing to consciously manage their scale and position, and noted that they were more accurate with point selection. An in-depth, detailed explanation of all of the findings and conclusions can be found in *Appendix 21*.

7.3. Experiment #3 (Scale Terrain)

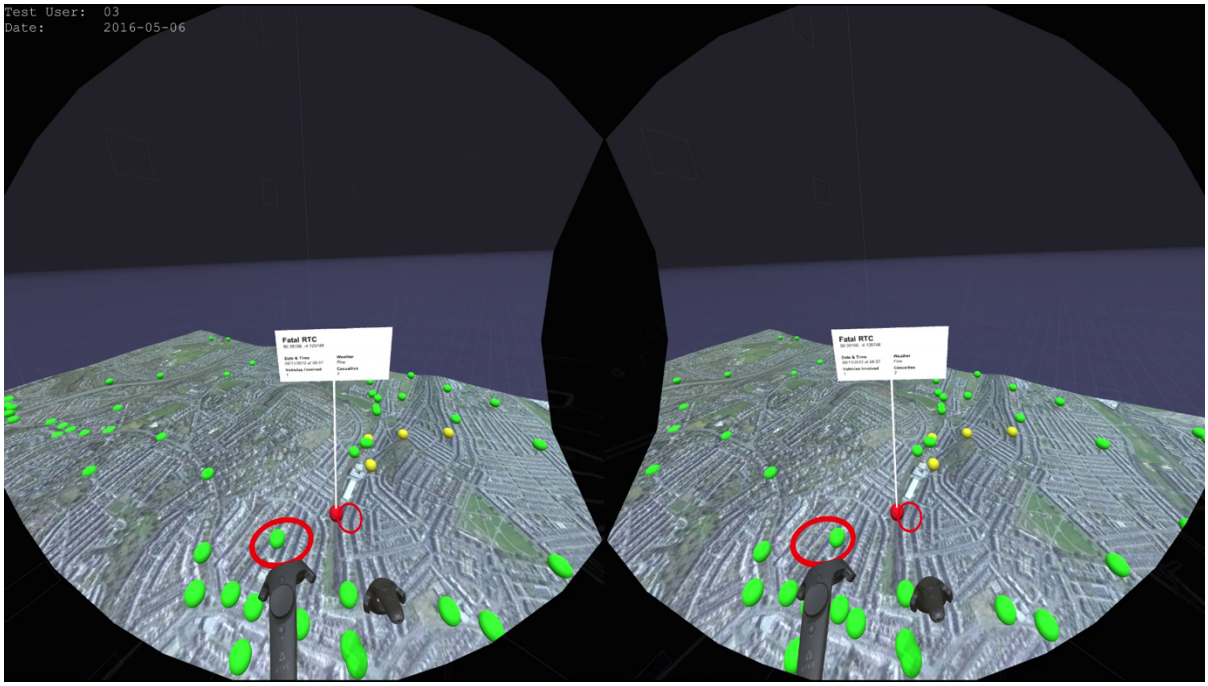

***Figure 6**: A screenshot of the final, VR version of Experiment #3. Note the halos on the controllers, the free-standing map with data points, and the dark environment.*

The table experiment was successful in creating a comfortable environment for viewing the data at distance, getting a general understanding of the scope and magnitude of the data set. The experiment was less effective at enabling the user to view data in detail, especially with dense data clusters.

With the final experiment, the user needed to both have a good overview of the data and also be able to understand and interpret detailed, dense data in a specific part of a map. Users wanted to be able to view more information, including information about multiple data points at once. The controller selection “halo” system was built to facilitate better selection of data points that could be either near or far away.

The final experiment also required the creation on a non-VR version of the experiment to be used as a control for user testing to see if the VR proof-of-concept is at least as effective for data visualisation and basic analysis as a traditional two-dimensional display interface (*Figure 6*).

7.3.1. Pan, Rotate, & Scale in RSVR

***Figure 7.1**: Panning or moving with the "Natural" Controller Gesture System*

***Figure 7.2**: Rotating with the "Natural" Controller Gesture System*

***Figure 7.3**: Scaling with the "Natural" Controller Gesture System*

Panning (*Figure 7.1*), rotating (*Figure 7.2*), and scaling (*Figure 7.3*) are intuitive, natural feeling gestures that show off the power and simplicity of multi-touch displays, such as those on the iPhone or iPad. In building a dynamic, responsive map system, it was clear a RSVR analogue of these gestures was required.

The project exposed many interesting implementation details and issues related to the "Natural" Controller Gesture System, which are expanded upon in depth in *Appendix 22*.

7.3.2. New Terrain System

While building the responsive controller system it became apparent that the Unity terrain system that had been used to create the map from the height map and its respective diffuse map was not built to be moved, rotated, and scaled in real time. The solution was to have a dynamic new terrain system that met the following requirements:

- It will create a mesh from a standard grayscale height map, and cover that mesh with the map's diffuse texture.
- The mesh will be moveable, rotatable, and scalable in real time.
- The new terrain system will work with large height maps and not result in frame rates dropping below 90 fps on the development machine.

A number of complex, technical issues arose during the development of the custom terrain system, outlined in *Appendix 23*.

The final new terrain system was able to generate a set of tiled meshes created from a single master height map, and a single master diffuse map. It was fully compatible with the responsive controller system (*§7.3.1* and *Appendix 22*), and worked up to several million vertices before bringing the development machine below acceptable frame rates. Early research indicates this could be elevated with less GPU-intense shaders on the terrain mesh.

7.3.3. Data Panels in the Virtual Environment

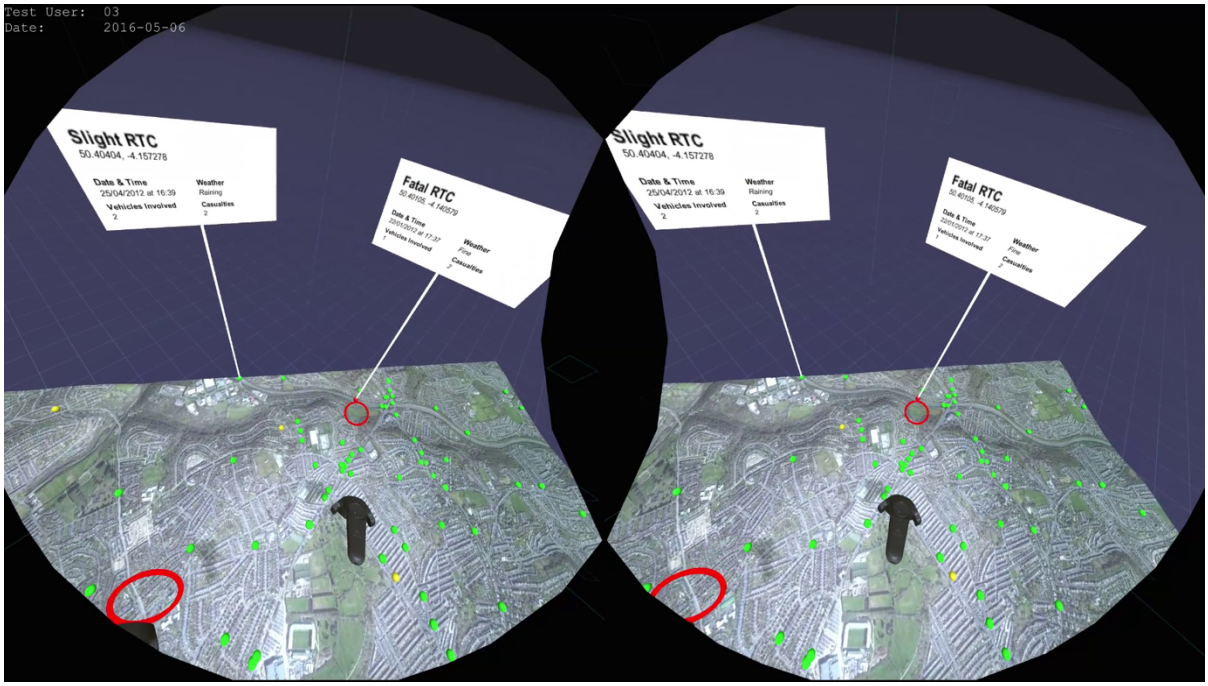

***Figure 8**: The data panels, now in the environment.*

Feedback from the previous experiment lead to the final experiment having data panels attached to their corresponding data points, and the option to have many data panels open at once to compare and contrast between data.

The new panels (*Figure 8*) float 1.2m above the data point, just below the average user's head height. Further research could investigate the optimum height for the panels, which could be placed dynamically, since the height of the user can be calculated (since the position on the Y axis of the HMD is known). That considered, the 1.2m height seems to work well for most users. The panel also rotates on the Y rotation axis to always face the user, meaning that a user never needs to move their body to read a panel, just turn their head.

The panels were also modified to be unlit by the environment, so there can provide the greatest contrast between the background (white) and the text (black). This, along with the larger typeface and more concentrated data mean that data panels are now readable from several metres away, so many more can be had side-by-side.

7.3.4. Extending & Contracting "Halo" Selection System

***Figure 9**: A diagram of the extending "Halo" selection system was an evolution of the data selector (Figure 4) that allowed for more control by the user by pressing the trackpad (big central circle). Note that the extended halo was actually 80cm from the controller, and not ~10cm as shown here.*

One of the issues moving from a table-based map interface to a more dynamic one that can move, be rotated, and be scaled, is that data points can be near or far from the user, and they can be as low as the ground, or as high as their torso. To build a selection system that was

comfortable to use, it had to be able to be both accurate with near and distant selections, and simple to interact with. It also had to be clear what the user was selecting, especially in dense data clusters.

The solution devised was a "halo" system (*Figure 9*), where-in a halo floats 5cm from the end of the controller, and can be extended to 80cm by pressing the controller's trackpad. The halo can be contracted by pressing the controller's trackpad again.

The halo has a collision trigger inside the ring which allows for selection of data points. If one or more data points is in the ring, it will select (or deselect) the one closest to the centre of the ring. This simplification from the multi-selections of earlier selection systems enables easy selection of different panels in dense clusters of data.

To show when an object is selectable, the ring can change colour when there is a selectable data point in its collision trigger. In the proof-of-concept, this changes the colour of the halo from red to green.

7.3.5. Adapting a VR Experience to a non-VR System

***Figure 10**: The non-VR mode of the software solution. It has buttons to switch to VR-mode.*

In order to test the merits of a VR system, a non-VR version was created to be used with control groups. The non-VR version of the proof-of-concept uses a third-party RTS Camera system (*§4.1.2*), which was modified to more accurately simulate how top-down, two-dimensional camera systems work in traditional geospatial data visualisation systems.

The non-VR version of the proof-of-concept (*Figure 10*) also required a modified data panel that looked directly up, and a new selection system. The new selection system worked by

having a panel enabled by clicking on its corresponding data point. The panel can be deselected either by clicking on the data point again, or by clicking on the panel itself.

It was also important to be able to switch between VR mode and the non-VR modes at runtime, without restarting the application or switching to a different application. The desktop application (the window on a regular monitor, as opposed to the view in the HMD) contains buttons to switch between the two modes, and even maintains panel selection between the modes.

7.4. User Testing

With recommendations as a deliverable, it was important to ensure the recommendations are well founded. A key part of the user-centered design approach is to keep a tight feedback loop with the users. Informal testing took part throughout the design and development process, but it was also important to have a large, encompassing user test through which the findings could be validated.

7.4.1. Objectives of User Testing

User testing was used throughout the process to ensure the project was moving in the correct direction. In the final stage of user testing, it was important to ensure that the research as a whole was moving in the correct direction, and that the ideas and recommendations provided in this project create a better data visualisation product.

As a result, the following objectives for user testing were laid out:

- Discover whether the user can move and engage with the virtual environment naturally, comfortably, and easily in RSVR.
- Discover whether users in the VR data visualisation can discover and comprehend a dataset faster, more accurately, and more enjoyably than the non-VR data visualisation interface.
- Discover whether the experience that a user has in a VR environment is dependent on their experience level with VR, and whether the user's physical differences alter their experience with the product.

From these, a user testing process was devised to satisfy the objectives.

7.4.2. User Testing Process

***Figure 11**: A user testing the non-VR build of the software solution.*

One of the key ideas the project was attempting to discover was whether VR interfaces can, in their infancy, provide a more effective interface to data visualisations than non-VR interfaces. As previously discussed (*§7.3.5*), a control application was developed to be tested against the primary VR proof-of-concept.

There were 9 users testing the VR proof-of-concept, and 8 users in the control group (*Figure 11*), testing the non-VR version. The users were selected largely from within the Computing and Mathematics School at Plymouth University, which could lead to a sampling bias skewing the findings (*Appendix 27*).

The users, regardless of their group, were given a questionnaire (*Appendix 24*), containing demographic questions and questions about their competency and experience with technology.

The testing process (*Appendix 25*), comprised of a brief verbal introduction to the control scheme. The application was then started, and the following tasks were asked of the user:

1. Identify a fatal Road Traffic Collision (RTC), select it, and say the date and time it occurred. Once they've completed the task they should deselect the data point.
2. Select two RTCs near the coast, and say the difference in the number of casualties, if there are any. Once they've completed the task they should deselect the data points.
3. Say the number of serious, but not fatal, RTCs.

The users were then thanked for their participation and dismissed.

7.4.3. User Testing Findings

See *Appendix 26* for an in-depth analysis of the user testing data, and *Appendix 27* for extended analysis on the reliability of the data.

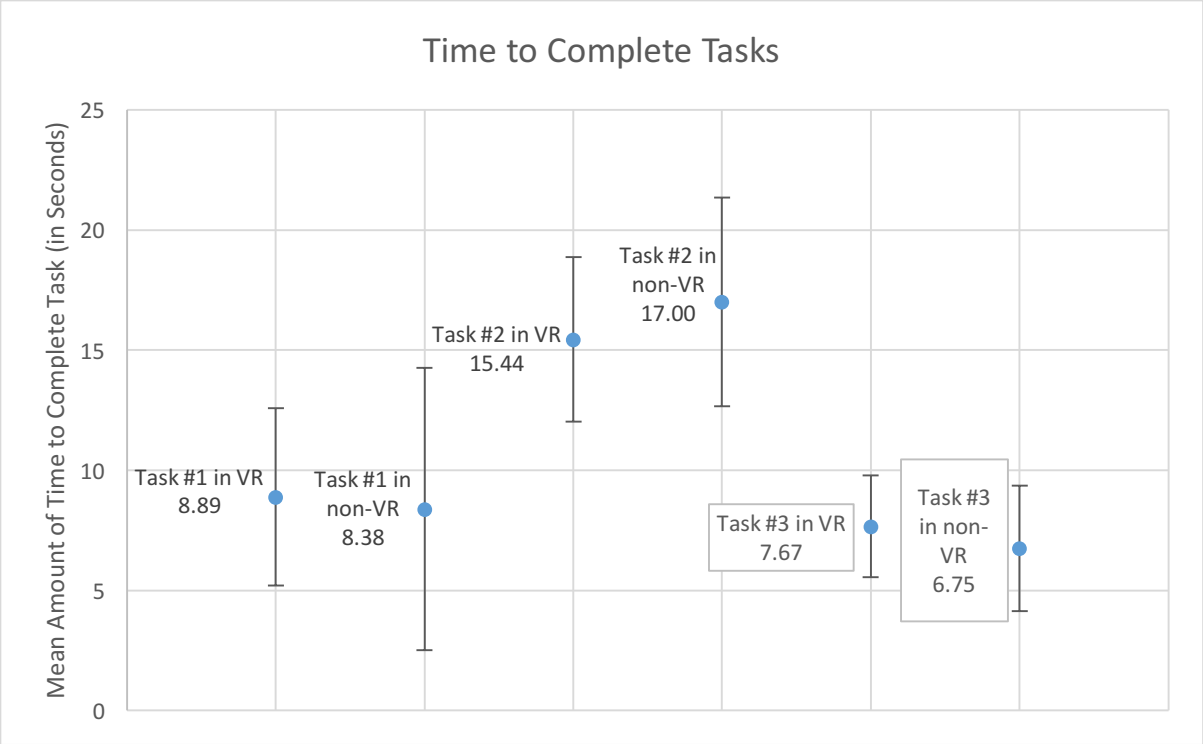

***Figure 12**: With standard deviation taken into account, the technology being used did not seem to have any significant statistical effect on the time to complete a task.*

Users in VR on average were only able to perform one of the three tasks faster, although it was the more demanding second task. Time to complete tasks were, with standard error taken into account, almost indistinguishable (*Figure 12*), and seemed to influenced more on the individual than the technology. With regard to accuracy, only one user made a mistake (in the non-VR system) in the three tasks, so no statistically significant conclusions can be drawn.

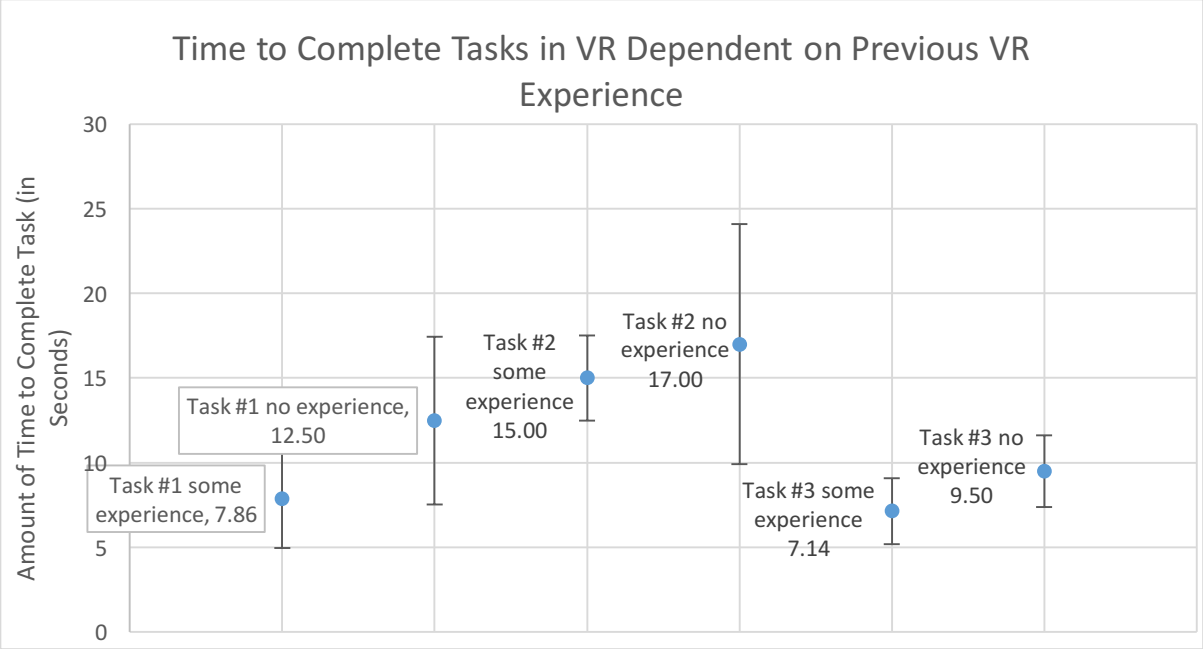

***Figure 13**: Users with previous VR experience were consistently able to perform better in VR tasks.*

Users who had previous experience in VR systems, regardless of their type (RSVR vs regular VR), consistently performed better in VR tasks than users who had not used VR in the past (*Figure 13*). However, the low number of users who used the VR solution as their first VR experience could be an anomaly.

Due to the small sample-size, definitive results cannot be drawn as to the viability of RSVR data visualisation, but it can be used in moderation to inform recommendations for future developers and research.

8. Recommendations

In this section there are a number of recommendations for designers and developers of VR software and platforms. The recommendations are formed based on experience building and testing experiments (*§7*), and understanding the literature that surrounds VR and data visualisation (*§3*).

These recommendations can be used by the creators of software and platforms to make consistent, comfortable, productive environments.

8.1. Room-Scale as an Interface for Data Visualisation

RSVR may not yet be entirely ready for the mainstream public, but the right moves are being made in the right direction to indicate it could not be long until visualisation software is predominantly used in virtual environments.

8.1.1. Speed and Accuracy

An important aspect of using data visualisation and analysis software is the speed and accuracy at which users can complete tasks. speed to perform tasks did not significantly decrease while using RSVR (*Figure 12*). It could be suggested that, based on the findings (*Figure 13*), users with past experience of VR systems might be quicker at performing tasks in RSVR than in traditional data visualisation interfaces. More research is needed to confirm these very early results.

No significant results were found to suggest that either VR or non-VR systems have a higher accuracy rate, since only one user mistake occurred during the entire test. Other tests can be performed in the future to focus on getting more actionable data for accuracy.

8.1.2. Potential for Pattern Discoverability

VR is the most immersive form of interface yet invented. Total immersion in a dataset investigation, with no distractions from the real world, maximises the user's possibility of discovering new and useful patterns in datasets.

Early user tests seem to suggest that users in VR perform as well as users in traditional systems in pattern discoverability (*Figure 12*, *Task #3*). The literature would also suggest that the higher field-of-view in VR could enable better pattern recognition (*§3.1.3*). Further, more in-depth research is required to confirm that the performance of pattern discoverability and recognition is indeed as good in VR as non-VR systems.

8.1.3. Observing & Interacting with Datasets

Arguably the key feature of a data visualisation program is observing and interacting with datasets. To enable a more performant user experience in VR, the user needs to be able to perform three things well in their virtual environment:

- Handle large amounts of data, and view data at a macro-level. This involves being able to quickly and easily manipulate the location and orientation of data sets as a whole. The user should be able to move around a dataset, seeing it from different angles, enabling them to discover trends and patterns. One of the key ways that users discovered patterns was through the use of colour. When used effectively, colour can enhance a dataset beyond just the position and density of its data points.
- Focus in on small clusters of data, at the micro-level. It is important to be able to move in and out of micro and macro levels of detail quickly and effectively. Once viewing the data at a micro-level, it is important to be able to move around it clearly, and select its data points accurately and reliably.
- Be able view and compare individual data points, at the atomic-level. While it is useful to view aspects of a dataset from a distance, it is also important to be able to view and compare individual data points. The key to presenting the data (as discussed in depth in *§3.2*) is to have high contrast, large typography in a consistent grid formation. Large clear text is vital to readability, due to the current low resolution of consumer HMDs relative to their field-of-view.

8.2. Interface Paradigms

Beyond interactions with data sets, it is important to be able to interact with the world in a virtual environment to build a space where the user is consistently comfortable. New user interface paradigms are required in order to enable users control over their entire environment, an ability they could never possess with prior technologies or interfaces.

8.2.1. "Natural" Controller Gestures

The author recommends the use of "natural" controller gestures (*§7.3.1*). These gestures, adapted from touch screen interfaces, provide the user with a paradigm they recognise from devices and interfaces they use day-to-day.

The gestures provide the user with instant feedback from even the most minor movements, meaning that based on only a moment's use, a user a recognise (and not need to recall) what the gesture does. The results of the user testing suggest that most users understand the "Natural" Controller Gestures, since most users (*Table 4*) used them to move their environment through the user tasks.

For further research, the author would like to see how users react to differing rates of scaling and panning based on the scale of the object they're interacting with. The proof-of-concept has a consistent rate of panning and scaling, but the author believes that the panning is too unresponsive at high scale rates, and the scaling is not responsive enough at low scale rates. An ideal rate curve has yet to be found for panning and scaling, but the author feels as though rotation works well at any scale.

Another potential avenue for research is based around the idea of momentum being added to objects by interactions, instead of direct manipulation. This concept works well on touch-

screen interfaces, so it could be interesting to see how users react to this in an immersive virtual environment.

8.2.2. Extendable Selection System

In early testing (*§7.1.3*) it was discovered that users don't like to reach extensively beyond their current position, especially downward. After testing bringing the data set up to the user (*§7.2*), the idea of an extendable selection system was proposed (*§7.3.4*).

Empowering the user to adjust their own reach distance enables them to move the controller less, exerting less energy, permitting them to remain comfortably in virtual environments for longer. It also reduces the need for adjusting the world's scale and position, allowing them to reach outside of the physical space that would otherwise limit them.

As a topic for further research, it would be interesting to see if more adjustment levels, or more control over variable-sized extensions could provide a better, more comfortable user experience.

8.2.3. Adapting to Different Users

A virtual environment, and the user interface within it, should adapt to the needs and requirements of individual users. The exponential increase in number of degrees of freedom for user input mean that it is important to consider all users, not just the 'model user'.

Control schemes should also be ambidextrous where possible. It is often impractical to know one's users' handedness beforehand, so control schemes should work equally well for left or right handed users. Users should be able to do any controller-based interaction with either controller, and ideally with only one controller, although that makes concepts like "natural" controller gestures difficult.

The software should work for users with different heights. A short user may only be 120cm tall, whereas a tall user may be as tall as 200cm (or even taller), and the interface should adjust to be comfortable for both users. This also affects users in wheelchairs.

Needless to say, users with physical disabilities should not be excluded from using RSVR software. Houldon (*2016*) elaborates (*§3.1.2*) how RSVR software can be designed to adapt for people of all sizes, shapes, and abilities.

8.3. User Wellness

In order for users to feel comfortable in VR, a number of additional factors must be met, spanning technical implementation details to ideas of emotional comfort.

8.3.1. Physical Comfort

There are a number of technical and creative decisions that can, and should, be made in order to provide the user with an acceptable level of physical comfort (*§4.4.2*).

The first of which is the prevention of nausea. Nausea is caused when the vision system and inner ear cannot agree on the user's position in space, and can cause great discomfort or even vomiting. To prevent nausea, the software application should not drop below 90fps (or 60fps with 120fps reprojection). This is explained in more detail in *§4.4.2*.

To enable extended usage, eye strain must be minimised. From informal testing, users self-report more comfort when using VR in a dark virtual environment. Further research could take place to test the validity of this hypothesis.

8.3.2. Emotional Comfort

In order to be able to be productive in a virtual environment, it is important to provide an environment that is free of perceived danger. This means that if possible, interface elements should avoid moving near the user's head and face, and sounds should not appear from behind.

Users should also be in large environments, if appropriate for the dataset. Even if they are in a large physical room, users will experience symptoms of claustrophobia in small or confined virtual environments.

To enable the user to focus their mental energy on analysing the data and finding trends and patterns, it is important that they are not needing to heighten their fear and alertness in order to respond to virtual threats to their safety.

9. End of Project Report

9.1. Project & Achievements

The project involved the creation and implementation of basic interface paradigms for RSVR. New platforms need consistent, comfortable paradigms to enable their users to work productively over long periods. The project was an early investigation into potential solutions for this need in RSVR.

A proof-of-concept was developed to experiment with, and demonstrate, the proposed recommendations. It enabled users to explore the Plymouth area overlaid with information from a RTC data set. The proof of concept was also ported to non-VR platforms to enable testing of control groups.

The recommendations span from technical implementation details to overarching goals that developers of VR software should strive for. The recommendations can be used by platform owners to develop early HIG documents to help their developers build user-centered products for their VR platforms.

9.2. Objectives & Requirements

Objectives are key to evaluating the success of a project. This project had a number of general objectives, and each of the deliverables had their own requirements and initial scopes. Both are discussed and evaluated here.

9.2.1. Analysis of Existing UI Paradigms

"*To analyse existing user interface paradigms for interacting with large volumes of three-dimensional data and provide recommendations for improvement"*

In the literature review (*§3.2*), information about existing interface paradigms is discussed and evaluated. Between the PID being established and the project beginning in earnest, however, the project's data visualisation goals shifted from large amounts of three-dimensional data to medium amounts of geospatial data. As a result, the literature review focuses primarily on geovisualisation interfaces and displaying visual information in a clear and balanced way, which were important to presenting the information clearly and without designer bias.

The research was primarily focused on how to present the data, since it was decided post-PID that the interface paradigms should be more related to the VR focus, instead of the data visualisation side of things.

9.2.2. User Personas & Stories

"To develop and maintain user personas and user stories in line with the Agile methodology of software development."

User stories and personas were developed early on in the process (*§2.3.2*). User personas did not need to be maintained since the user the software proof-of-concept was for did not change during the development process.

User stories were maintained during the development process, and were used to ensure that features required to meet the baseline of both user and functional requirements were met. These were updated in Pivotal Tracker as the project went on, and served as a development timeline of features being identified, developed, and finished.

Because the agile methodology of software development was not maintained throughout the project (*§10.2*), and the project moved toward an exploratory and experimental direction, user personas were not as important during the project as it was expected they would be when developing the PID.

9.2.3. Software Proof-of-Concept

"To develop a core interface to datasets in VR in line with the initial scope."

The final software proof-of-concept needed to fulfil the requirements set out in the initial scope of the PID. An evaluation of the completion of each item of the initial scope can be found in *Appendix 28*.

This objective was completed, although in a modified state. The objective made the assumption that changing the scale of the user was the solution to making software that scaled from small datasets to large ones, but my research (*§7.1.3*) showed that in order to prevent nauseating experiences, it was better for the user to directly manipulate their environment. With that considered, the objective was completed.

9.2.4. Proposals & Recommendations

"To develop a set of proposals for future data visualisation software using RSVR systems."

The proposals, demonstrated in *§8*, are a set of proposals for applicable interface paradigms for RSVR. The recommendations go in-depth discussing what works well in VR, proposals for interesting paradigms, and potentials for further research needed for building a full HIG document.

9.3. Evaluation of Objective Completion

As discussed in *§10.1*, despite all objectives being completed, the first objective regarding research was modified after the creation and submission of a PID due to changes in the requirements as the project progressed.

10. Project Post-Mortem

A project post-mortem enables the author to report on how the project went, and discuss how he can learn from the successes and failures of the project to improve for the next project.

10.1. Objectives

Since objectives are central to the success of a project, it is important, after a project, to evaluate the objectives themselves. We review the objectives to evaluate their appropriateness and identify areas where improvements could have been made.

As discussed in *§9.2.1*, the original data visualisation objectives did not reflect the eventual visualisation approach. A better objective would've been to collect and display research on how existing data presentation techniques are used. At the beginning of the project, the focus was on representing large three-dimensional clouds of data. Subsequently, the project pivoted to representing geospatial data on map-like interfaces. Because of this shift the specifics of the objective changed, but the overall nature of it, to research non-VR data visualisation and presentation techniques, stayed the same.

The user personas and stories objective (*§9.2.2*), was closely tied to the successful completion of the agile process. Because the project was not a perfect implementation of the agile process, a better objective might have been a little more generalised, such as:

"To implement, and document any changes to, a modified version of the Agile software development process."

This alternative objective would have allowed more leeway in the event that the process was adapted to fit the needs of a single-person team developing experimental software (*§10.2*).

The objectives for the software application (*§9.2.3*) were well written and useful when developing the software. If I were to write them again, I would focus on writing user requirements rather than functional requirements as objectives. Functional requirements change and are adaptable to changing data sets, but user requirements often are less fluid and resistant to changing circumstances.

The recommendations' objectives were also well written, and allowed me enough space to experiment with interface paradigms without having to conform too closely. Being able to learn through experimentation and then report the findings of the experiments in the form of recommendations was valuable as a learning exercise, and produces value for the community at large, as knowledge creation and curation.

10.2. Development Process

The Agile / Scrum software development process was chosen because it enables rapid, iterative development of a product with changing goals. Initially I expected the

experimentation phases to follow a more regimented flow, fitting nicely into two-week sprints. I understand now that the project was less a development project, which would have been an implementation of ideas, and more of a design project. With this knowledge, I would approach the project again using something similar to the GV Design Sprint (*Knapp, 2016*), which puts emphasis on coming up with ideas and testing them quickly. This would have enabled me to generate ideas and rapidly identify optimum solutions. In turn, this would have given me more time to polish the *right* experiment for the final proof-of-concept software solution.

Project meetings with the supervisor were useful, but did not fit into the Agile approach as much as I had hoped they would. After the initial sprint reviews, true reviewing (going through and accepting completed stories as they are demonstrated) didn't take place as often or as thoroughly as I would have liked. In retrospect, this aspect was one of the weakest parts of the Agile approach during the project.

It was discovered early in the project that formalised standup meetings on a single-person project were a waste of time, since in group projects they are used to ensure everyone on a team is on the same page, and on a single-person project this is not an issue. I hypothesised that lonesome stand ups would enable me to develop strong, daily to-do lists, but this ended up not being the case, and after a week the formalised stand ups were removed. Further analysis of agile in a one-person team can be found in *Appendix 29*.

One major short-fall of the software development process on my part was that I did not commit as often or as selectively as I ought to have. In experimental development, it is important to have a strong, stable commit history to revert back to if need-be. If I knew that experiments were going to be as clear, and as key, as they were, I would have probably used Git Flow, which provides a greater structure around releases and feature development. Git Flow would have enabled me to siphon off releases into stable release branches to be referred back to. As it currently stands, if one wanted to return to a stable version of an experiment, one would need to revert the entire repository back to its merge commit.

10.3. Technologies

Technologically the proof-of-concept application is well built, using Unity as the game engine, and C# as the language for the scripts and modules. The reasoning for using Unity and C# was strong at the beginning of project, with a good workflow for developing prototypes quickly, a decent VR plugin environment, and an excellent community of designers and developers who have created a treasure-trove of documentation and debugging assistance.

Working with the HTC Vive in its pre-release state was very difficult. The early hardware was flakey, its firmware was unreliable and inconsistent, and the frameworks for building VR software in Unity were largely undocumented. Developing on an unreleased platform was very interesting from a design point-of-view, since I had creative freedom to try new things

on a blank canvas, with few established design paradigms, but was significantly more difficult to develop on.

In order to understand how the controller mechanism worked, I had to delve deep into the source code to the point where the framework was interacting directly with the OpenVR drivers. I had to modify the framework to expose features that were not publicly accessible in order to get basic functionality, such as reliably tracking the order and position of the HTC Vive Controllers. It is clear to me now that working on the cutting edge of technology is very difficult, but very rewarding.

10.4. Conclusions

I am very proud of both the software proof-of-concept I created, and the recommendations developed. I firmly believe that the work I have produced is valuable knowledge to enable developers and platform owners to put systems in place to best provide for their users. This early design and prototyping work is an important and vital first step in developing and establishing a consistent platform that users are comfortable and confident using.

A modified project management approach, better tailored to a sole-developer, would have enabled me to discover and iterate upon ideas quickly. This faster turn-around could have resulted in discovering more ideas, giving me a better base to establish my recommendations upon.

If I could do it again, I would most likely use a more mature platform, and focus on either creating experimental experience demos, or building a set of thorough recommendations. Instead I opted to do both, and both deliverables suffered for having less focus. I would focus on developing in-depth interactions and paradigms for more mature platforms (with well documented frameworks and workflows), and focus on developing those interactions for a particular productive niche.

References

1. Barrett, J. (2003). *Minimising side effects of virtual environments* (No. DSTO-TN-0478). Defence Science and Technology Organisation, Info Sciences Lab. Salisbury, Australia.
2. Bowman, D., Kruijff, E., LaViola, J. and Poupyrev, I. (2001). *An Introduction to 3-D User Interface Design*. Presence: Teleoperators and Virtual Environments, 10(1), pp.96-108.
3. Chesher, C. (1994). *Colonizing virtual reality: Construction of the discourse of virtual reality, 1984-1992* (Vol. 1, No. 1). Cultronix.
4. Department of Transport (2015). *Road Safety Data*. [data file]. Available at: https://data.gov.uk/dataset/road-accidents-safety-data [Accessed 28 Apr. 2016].
5. Dickson, W.P. (2005). *Toward a deeper understanding of student performance in virtual high school courses: Using quantitative analyses and data visualization to inform decision making*. A synthesis of new research in K–12 online learning, pp.21-23.
6. Dykes, J., MacEachren, A.M. and Kraak, M.J. (2005). *Exploring geovisualization*. Elsevier.
7. Freina, L. and Ott, M. (2015). *A literature review on immersive virtual reality in education: state of the art and perspectives*. In The International Scientific Conference eLearning and Software for Education (Vol. 1, p. 133). " Carol I" National Defence University.
8. *Geovisualisation*. (2011). 1st ed. [ebook] Corvallis, Oregon, USA: Oregon State University, pp.169-174. Available at: http://dusk.geo.orst.edu/gis/Chapter13_notes.pdf [Accessed 30 Apr. 2016].
9. Git (n.d.). *The Git Version Control System*. [online] Available at: https://git-scm.com/ [Accessed 5 May 2016]
10. Heilig, M. (1962). *Sensorama Simulator*. U.S. Patent No. 3,050,870.
11. Herndon, K., van Dam, A. and Gleicher, M. (1994). *The challenges of 3D interaction*. SIGCHI Bull., 26(4), pp.36-43.
12. Houldon, S. (2016). *VR Specific Accessibility Thoughts*. [online] Available at: https://workflowy.com/s/OgHCXki3ZE [Accessed 29 Apr. 2016].
13. HTC, (2015). *HTC and Valve Partner To Make Virtual Reality Dream Come True*. [online] Available at: https://www.htc.com/us/about/newsroom/2015/2015-03-01-htc-valve-partner-to-make-virtual-reality-come-true/ [Accessed 28 Apr. 2016].
14. Knapp, J. (2016). *Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days*. 1st Edition. Simon & Schuster.
15. MacEachren, A., Gahegan, M., Pike, W., Brewer, I., Cai, G., Lengerich, E. and Hardistry, F. (2004). *Geovisualization for knowledge construction and decision support*. IEEE Comput. Grap. Appl., 24(1), pp.13-17.
16. Machkovech, S. (2015). *SteamVR: The room-scale VR world that feels like an "IMAX in your house"*. [online] Available at:

http://arstechnica.co.uk/gaming/2015/06/steamvr-the-room-scale-vr-world-that-feels-like-an-imax-in-your-house/ [Accessed 5 May 2016].
17. Martin, R.C. (2003). *Agile software development: principles, patterns, and practices*. Prentice Hall PTR.
18. Ni, T., Bowman, D. and Chen, J. (2006). *Increased display size and resolution improve task performance in Information-Rich Virtual Environments*. In: Graphics Interface 2006. Toronto, Canada: Canadian Information Processing Society, pp.139-146.
19. Oculus (2016). *Simulator Sickness*. Oculus Developer Center. [online] Available at: https://developer.oculus.com/documentation/intro-vr/latest/concepts/bp_app_simulator_sickness/ [Accessed 5 May 2016]
20. Oculus VR, (2012). *New Virtual Reality Gaming Headset From Oculus VR™ Gets Kickstarted*. [online] Available at: https://www1.oculus.com/press/new-virtual-reality-gaming-headset-from-oculus-gets-kickstarted/ [Accessed 28 Apr. 2016].
21. Ohannessian, K. (2015). *The Technical Challenges of Virtual Reality*. [online] Available at: http://iq.intel.com/the-technical-challenges-of-virtual-reality/ [Accessed 5 May 2016]
22. Opensource.org. (2016). *The MIT License*. [online] Available at: https://opensource.org/licenses/MIT [Accessed 29 Apr. 2016].
23. Otelsberg, J., Akshay, N. and Bhavani, R.R. (2013) *Issues in the User Interface Design of a Content Rich Vocational Training Application for Digitally Illiterate Users*. International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering, 7(10).
24. Pritchard, T. and Radak, J. (2016). *1: The Train to El Dorado, with Joe Radak of Eerie Bear Games*. [podcast] Immersive. Available at: http://www.immersivepodcast.com/episodes/1 [Accessed 28 Apr. 2016].
25. Schwaber, K. (1997). *Scrum development process*. In Business Object Design and Implementation (pp. 117-134). Springer London.
26. Snow, J. (1855). 1st Ed. *On the Mode of Communication of Cholera*. London, England.
27. Stack Exchange (2013). *Version Control for Game Development*. [online] Available at: http://gamedev.stackexchange.com/questions/480/version-control-for-game-development-issues-and-solutions [Accessed 5 May 2016]
28. Sutherland, I. (1963). *Sketchpad, A Man-Machine Graphical Communication System*. Ph.D. Massachusetts Institute of Technology.
29. Swink, M. and Speier, C. (1999). *Presenting Geographic Information: Effects of Data Aggregation, Dispersion, and Users' Spatial Orientation*. Decision Sciences, 30(1), pp. 169-195.
30. Sylkin, D. (2016). *RTS_Camera*. [online] GitHub. Available at: https://github.com/densylkin/RTS_Camera [Accessed 28 Apr. 2016].
31. Tanriverdi, V. and Jacob, R.J.K. (2001). *VRID: A Design Model and Methodology for Developing Virtual Reality Interfaces*. In: VRST '01. Banff, Canada.
32. Tufte, E.R. and Graves-Morris, P.R. (1983). *The Visual Display of Quantitative Information* (Vol. 2, No. 9). Cheshire, CT: Graphic press.

33. UC San Diego. (n.d.). *How to Lie, Cheat, Manipulate, and Mislead using Statistics and Graphical Displays* [presentation] Available at: https://cseweb.ucsd.edu/~ricko/CSE3/Lie_with_Statistics.pdf [Accessed 7 May 2016]
34. Udod (2013). *Can you recommend a better IDE for Unity C# coding?*. [online] Available at: http://stackoverflow.com/questions/16984392/can-you-recomment-a-better-ide-for-unity-c-sharp-coding [Accessed 5 May 2016]
35. UK Government: National Archives. (2016). *Open Government Licence*. [online] Available at: http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/ [Accessed 28 Apr. 2016].
36. Valve Software, (2016). *OpenVR*. [online] GitHub. Available at: https://github.com/ValveSoftware/openvr [Accessed 28 Apr. 2016].
37. Van Lamsweerde, A. (2009). *Requirements engineering: from system goals to UML models to software specifications*.
38. Van Wijk, J.J. (2005). *The Value of Visualization*. In Visualization, 2005. VIS 05. IEEE (pp. 79-86). IEEE
39. Vink, P. (2005). *Comfort and design*. Boca Raton: CRC Press.
40. Vlachos, A. (2015). *Advanced VR Rendering*. [presentation] Available at: http://media.steampowered.com/apps/valve/2015/Alex_Vlachos_Advanced_VR_Rendering_GDC2015.pdf [Accessed 5 May 2016]
41. Volpe, J. (2016). *HTC's making virtual reality safe for the home with Chaperone*. [online] Available at: http://www.engadget.com/2016/01/05/htc-vive-virtual-reality-chaperone/ [Accessed 5 May 2016].
42. Von Krogh, G., Haefliger, S., Spaeth, S. and Wallin, M.W. (2012). *Carrots and rainbows: Motivation and social practice in open source software development*. MIS quarterly, 36(2), pp.649-676.
43. VR Devs (2016). *VR Devs Slack Group*. [slack group] Available at: https://vrdevs.slack.com/ [Accessed 5 May 2016, Log in required]
44. Wiederhold, B.K., Gevirtz, R. and Wiederhold, M.D. (1998). *Fear of flying: A case report using virtual reality therapy with physiological monitoring*. CyberPsychology & Behavior, 1(2), pp.97-103.
45. Woodson, W., Tillman, B. and Tillman, P. (1992). *Human factors design handbook*. New York: McGraw-Hill.

Appendix 1: Background on Human Interface Guidelines

Human Interface Guidelines (HIGs) have been important to establish a set of standard interactions between users and computer programmes (*Woodson, Tillman, and Tillman, 1992*). Their recommendations span a range of elements and paradigms that, once established, users can recognise and understand without having to develop a new mental model for interface elements on a per-application or per-experience basis.

HIGs are informed by research and early experimentation by software developers and user experience designers. The researchers test various interface metaphors, adapt existing paradigms for the new technology, and invent new paradigms and elements.

Appendix 2: Background on Room-Scale Virtual Reality

VR is a type of human-computer interaction identified by an immersive display seemingly placing the user in a virtual environment (*Freina and Ott, 2015*). RSVR is a new kind of VR that allows the user to move around in a room-sized physical environment (*Machkovech, 2015*). The user's physical movements are then translated directly to the virtual environment.

This project takes place in the early research stage of RSVR user interfaces. It involved experiments with new user interface paradigms, as well as porting existing interface paradigms to new environments. The project's recommendations could be used by teams developing new platforms and experiences based on similar RSVR technology to develop comfortable, consistent, and relatable experiences for their users.

Software developers and game developers building experiences for RSVR are having to explore new ideas and paradigms unique to the technical implementations and inherent issues of building tools in virtual environments (*Pritchard and Radak, 2016*).

Appendix 3: Background on Data Visualisation

Data visualisation is a vital component to understanding and beginning to make large-scale actionable decisions on data sets (*Dickson, 2005*). Data visualisation led the personal computer revolution, with Microsoft Excel bringing computing to the general workforce. In the next stage of personal computing, virtual and augmented reality, enhanced data visualisation and comprehension will be leading reasons for businesses to adapt their processes to incorporate new technologies and methods.

This project focuses on one type of data visualisation in-particular: geovisualisation. Geovisualisation is the visualisation of data with spatial relationships (*Dykes, MacEachren, and Kraak, 2005*).

Appendix 4: PID Objectives

The project objectives, set out in the Project Initialisation Document (PID), were designed to provide realistic core evaluation criteria for the project as a whole. In the PID, the objectives were laid out as follows:

1. To analyse existing user interface paradigms for interacting with large volumes of three-dimensional data and provide recommendations for improvement.
2. To develop and maintain user personas and user stories in line with the Agile methodology of software development.
3. To develop a core interface to datasets in virtual reality in line with the initial scope.
4. To develop a set of recommendations for future data visualisation software using room-scale virtual reality systems.

In order to have a valuable set of proposals for software developers and interface designers, they had to meet the following objectives:

1. The software is easy for first-time users to pick up and understand.
2. The software can be used for observing large amount of data, as well as individual data points.
3. The software can have a standard set of tools (similar to the WIMP—Windows, Icons, Menus, Pointers—system in mouse-based interfaces) for interaction with the software in virtual reality.

If the software proof-of-concept met these requirements would be considered a success.

Appendix 5: HIG Requirements

Virtual reality, and especially room-scale virtual reality, is a young industry, and polished, commercially available software is starting to appear on the various virtual reality platforms. The recommendations could be used by these early developers to build comfortable, impressive, and usable products and experiences.

The recommendations had the following requirements:

1. The recommendations needed to discuss the advantages and limitations of using software in virtual reality, and specifically, in room-scale virtual environments.
2. The recommendations needed to outline a set of proposals for interactions in room-scale virtual reality between a user and a large amount (>100 data points) of data. These proposals would need to address the needs of comprehending large amount of data from distance, as well as interacting with lone points of data.
3. The recommendations needed to address core ideas central to interfaces, such as changes in contexts, such as moving from observing a wide overview to a focused cluster of data, or selecting a single data point to view detailed information about it.
4. The recommendations also needed to briefly discuss how to ensure the user is both physically and emotionally well whilst interacting in a virtual environment.

These requirements were established in order to provide a framework for the areas of experimentation in the software project, outlining the particular sections of data visualisation this project investigated.

Appendix 6: Software Deliverable Explanation

The software application is delivered with both an executable and the source code, including all art assets and datasets. The application is only available on Windows as a result of the current platform-exclusivity of the OpenVR runtime. In order to use the VR component of the software application, the PC running the application must be connected to HTC Vive VR hardware, and have the latest beta of the SteamVR software runtime installed.

Appendix 7: User Requirements & Personas

The following key user stories shaped the development of the software product:

1. As a user, I want to be able to see data sets containing locations overlaid on a map.
2. As a user, I want to be able to interact with a data point to see more of its information.
3. As a user, I want to be able to understand data sets from a distance to discover and identify trends, but also have fine-grained control over individual data points.

The model user, or the "persona", is a data-scientist at a PhD level. They have to sift through large amounts of data to discover patterns. At the moment they are frustrated by how their data visualisation tools (running on a traditional Windows PC) seem to get in their way. They are annoyed by the confusing navigation controls, and want to be able to have a clearer view of all of their data, without the cruft of clunky navigation and menus.

Appendix 8: Functional Requirements

The user requirements and personas formed the basis for the initial functional requirements:

1. The user needed to be able to load a dataset into the software to be visualised. This dataset would then be shown on a map to provide context for the user.
2. The user needed to be able to move around the virtual environment in a natural way, allowing them to view the dataset and its surrounding context from multiple angles and viewpoints.
3. The users needed the ability to manage and control their scale and position in relation to the dataset, in order to be able to accurately observe and interact with data in both macro and micro levels.
4. Users needed to be able to interact with data points, allowing them to view information about it.

Appendix 9: The Open Government License

The Open Government Licence is a very open and flexible licence for distributing public sector information and data. It allows its licensees to distribute the information and adapt it

for both commercial and non-commercial use. The only caveat and requirement of the licence is that attribution of the source of the information must be included in the application. The application contained an ‘always-on’ label in the traditional display application.

The rights under this licence allow the project to include the modified dataset to be distributed in both the application binary and the source code, with proper attribution. Careful notice has been taken to ensure all of the appropriate attribution is given in all of the deliverables.

Appendix 10: Included Open Source Project License Compliance

The project includes other open source code, from OpenVR (for the VR base framework), and RTS Camera (for the non-VR camera system). Their open source licenses needed to be complied with.

OpenVR utilises a custom license, similar to the MIT license, that allows both source and binary redistribution, provided that the copyright notice and license are included in those distributions. This licence is included with both the binary and the source code of the software application.

RTS Camera uses the MIT Open Source License, which is very similar to the open license that OpenVR uses. It also permits both source and binary redistribution, with the caveat of including the MIT license with both distributions. The software project includes the license in both the binaries and the source code.

Appendix 11: Edits Made to Dataset

It is important that data is represented fairly, as explained in *§4.2*, so any modifications made to any datasets must be made with a conscious effort to minimise any introduced bias.

The data set from the Department of Transport originally came with every reported RTC in the United Kingdom, which included 145,571 collisions. To improve loading times, the dataset was trimmed to only include the collisions in the range of latitude and longitude of the Plymouth map area, reducing the number of data points to less than 400.

The data originally came in a format similar to *Table 1* (which has been truncated on both the x-axis and the y-axis for brevity). This information was difficult to present in its found-form, so the fields were modified to show the item from the key that the field’s data point represented. This resulted in the modified data set in *Table 2* (also truncated for brevity).

Accident_Index	Location_Easting_OSGR	Location_Northing_OSGR	Longitude	Latitude	Police_Force	Accident_Severity
201201BS70001	527200	178760	-0.169101	51.493429	1	3

201201BS70002	524930	181430	-0.200838	51.517931	1	3
201201BS70003	525860	178080	-0.188636	51.487618	1	3
201201BS70004	524980	181030	-0.200259	51.514325	1	3
201201BS70005	526170	179200	-0.183773	51.497614	1	3

***Table 1**: Data as it came in from the Department of Transport (truncated).*

Index	Longitude	Latitude	Severity	Vehicles Involved	Casualties
201250EK2A002	-4.120071	50.349221	Slight	1	1
201250EK2A012	-4.113625	50.353152	Slight	1	1
201250EK2A007	-4.112895	50.353175	Slight	1	1
201250EK2A015	-4.112828	50.353257	Slight	1	1
201250EK2A011	-4.111891	50.354317	Slight	2	1
201250EC2D018	-4.161466	50.362149	Slight	1	3

***Table 2**: Data with data replaced by information from keys, for example the severities have been changed from '1' to 'Slight', as outlined in the 'DoT Safety Data Guide'.*

Because no new information has been introduced, data has just been substituted from the keys (in the data guide) to the main dataset, and data from other areas has been removed, the information for the Plymouth area is still as reliable and fair as the original data set.

Appendix 12: Physical Safety in Tethered Virtual Reality Systems

The HTC Vive uses both its camera and its lighthouses to help users understand where they are in their physical environment, and warn them when they are approaching the edge of their 'play space' or an obstacle in that space (*Volpe, 2016*). To ensure the user is best prepared to safely interact with their physical environment in a virtual world, the room setup process must be properly completed, including setting up accurate boundaries. To allow the camera to accurately discover, and warn the user about, obstacles in their path the physical environment must be well lit in a uniformly white or incandescent light.

Despite the power of the HTC Vive's camera-based chaperone system, Volpe (*2016*) also notes that a clear and open room is required for room-scale virtual reality to be safe. Obstacles, such as boxes or chairs, can be tripped over causing personal injury and potential damage to the equipment. If the environment is clear and open, it must also not be too dynamic. Environments with moving objects, such as vehicles or pets, require the user pay attention to them. Trying to operate software in a virtual world will cause the user to not pay the appropriate attention to risks and dangers in the physical world.

Appendix 13: Comfort in Virtual Environments

In a virtual world anything can happen, including things that could not physically happen in the physical world. To ensure that the user is comfortable in virtual reality it is important to keep them from feeling in danger, since fear responses in virtual environments are very similar to fear responses in physical environments (*Wiederhold, Gevirtz, and Wiederhold, 1998*). Anecdotally, objects close the user's face or torso appear to make users consistently feel uncomfortable, as do fast moving objects near the user. To ensure the fear response is as low as possible, the virtual environment should be consistently safe, non-threating, and predictable.

Physical comfort in virtual reality is also important. The most prevalent issue that has plagued virtual reality research for decades has been motion sickness, which is sometime referred to as cyber-sickness (*Barrett, 2003*). These issues have largely been solved with the HTC Vive hardware as a result of two major changes:

1. All movement is initiated by the user, and tracked one-to-one in both the physical and virtual environments. This means that there is not a discrepancy with the eye's perceived movement, and the physical movement detected by the vestibular system (in the inner ear), which causes the traditional motion sickness often felt in early virtual reality systems (*Barrett, 2003*).
2. The display updates at ninety frames per second (90 Hz), which lets the eye perceive its movement at the same frequency it does in the real world (*Ohannessian, 2015*). In a lower frequency system head movement results in the vision lagging behind, causing a discrepancy between the eyes and the inner ear.

In order to minimise the risk of the user experiencing motion sickness the user must initiate all movements in virtual environments, and the virtual head must move as tracked, one-to-one, with the real world (*Oculus, 2016*). The software solution must also stay consistently above 90 fps, as dips under that framerate is jarring and disorienting.

With the display covering the entire eyesight of the user the entire time the user is in the virtual environment, it is important to ensure that the hardware and software reduce any strain on users' eyes. The hardware does this by only flashing the frame for a small amount of time and being off most of the time (*Vlachos, 2015*). To provide a comfortable environment for extended periods, the virtual environment should be dark if possible, as well.

Another way to reduce eye strain in head-mounted display systems without eye tracking and foveated rendering is to have users only focus on objects further than approximately 30 cm from their eyes. Focusing on objects closer than that cause the eyes to look at the edges of the HMD lenses, causing visual banding and chromatic aberration, which when viewed for prolonged amounts of time, can cause discomfort.

Appendix 14: Managing Risk as a part of Project Management

Risk management took place at the beginning of the project, and is outlined in detail in the PID. Four major issues presented themselves:

1. A difficulty implementing the loading and parsing of datasets.
2. An overrun of the sprint schedule.
3. A hardware failure on the development PC, or of the HTC Vive developer kit system.
4. A software failure.

An analysis of these risks can be found in the End-of-Project Report in *§9* and the Project Post-Mortem in *§10*.

Appendix 15: Reasons for using the Unity Game Engine

Unity was selected as the software environment to develop the proof-of-concept for three reasons:

1. The author has a significant amount of experience with the Unity game engine, shipping a number of projects over several years. A solid understanding of both the software application as well as the engine's API implementation meant that less time was required to learn the basics of the game engine required to meet a baseline requirements. Because of this, more time could be spent implementing the features that provided the value and new knowledge in the project. The author has experience with other game engines that could have been suitable for a project such as this one, but none to the degree he has with Unity.
2. There was an existing virtual reality workflow existing for OpenVR software in the Unity game engine. Valve Software, creators of the software for the HTC Vive, created and maintained a Unity plugin, "SteamVR", which enabled fast and efficient prototyping of virtual reality software in the Unity game engine. The ease of using pre-build prefabs in early experiments meant the author could quickly iterate on ideas while fleshing out the particulars of the features.
3. The Unity game engine has a large number of game and software developers building virtual reality content for it. This large and active community meant that when an issue was found, there were a lot of available and eager developers with experience in the same software suite. The Slack channel, VR Devs (*2016*), was an invaluable resource in developing software for the pre-release HTC Vive.

It should also be noted that Unity provides the opportunity to port the application easily to other platforms, such as Mac or Linux (when they receive OpenVR support), or even the PlayStation 4, with its PlayStation VR virtual reality system. These ports are out of the scope of this project, but could be an interesting opportunity for a follow-up project that is enabled by an open and portable game engine such as Unity.

Appendix 16: Reasons for using the C# Programming Language

C# was chosen as the primary development language for the project for the following reasons:

1. The author has extensive experience developing large pieces of object-oriented software using C# and the Microsoft frameworks (such as .NET) that surround it. The author also has experience designing and building games and entertainment experiences using the Unity game engine with the C# language. Using a known language enabled the author to focus on designing new paradigms and solving difficult technical issues as opposed to learning the syntax of a new language.
2. The C# language is very mature, and used in production environments around the world for enterprise and consumer facing applications, and so it has an extensive documentation, a deep knowledge-base, and vibrant debugging communities. With both the Microsoft Developer Network (MSDN) and StackOverflow's C# community, issues relating to the language or the core frameworks were likely to have been encountered and solved before by others.
3. C# is also tightly integrated with the Visual Studio IDE, widely considered to be the best IDE available for developing with the Unity game engine (*Udod, 2013*). Advanced intellisense (intelligent code auto-completion) and robot debugging tools make C# with Visual Studio an efficient and effective combination for rapidly building and iterating on prototypes.

Appendix 17: Laboratory & Physical Environment

***Figure 14**: The VR lab used in Plymouth University. It has an active play-space of 3.2m by 2.9m, and can be almost completely soundproofed, due to its curtain walls.*

RSVR requires a room-sized area, and as a result the project required consistent access to a private, lockable, room of sufficient size. This was a non-trivial ask of the faculty, but Babbage Room 201, the "VR Lab" (see *Figure 14*), was available for the semester, so was used as a development and testing area.

The VR Lab has a 'playable space' of about $3m^2$. It was cleared and set up to be consistent with the health and safety requirements laid out in *§4.4.1*.

Appendix 18: Converting Latitude & Longitude to Cartesian Coordinates

Technically, implementing a conversion from latitude and longitude coordinates in the dataset to the Cartesian coordinates used in the Unity game engine was difficult, due to the complexities of mapping spherical positions to two-dimensional maps. In the end an approximate solution was found that worked well (with an acceptable margin of error) with maps less than $100km^2$, using the north-west most and south-east most coordinate of the map. This does not take into account the curvature of the earth, resulting in a greater error closer to the centre of the map, but on maps the size of the Plymouth one it works well.

Appendix 19: Extended Conclusion for Experiment #1

Users first reported was that they felt motion sickness scaling themselves from their heads, but this issue seemed to disappear when scaling the user from the estimated position of their feet (which had to be calculated based on their head position, and the floor's position, since the feet are not tracked in the HTC Vive system).

Users also did not like having to be cognitively conscious of managing their own scale. One user found themselves in the wrong part of the map, and unable to walk to the other part of the map (the room was too small at the user's current scale), they tried to scale up to make the map smaller, but due to their position within the room, the part of the map they were trying to reach was still out of the physical boundaries of the room-scale system.

It was also reported that reaching down to interact with data points, which were on the ground, was difficult, and potentially uncomfortable. Users did not have to reach down, since the laser-pointer worked at any distance, but accuracy of the pointer decreased as the distance from the floor increased.

Finally, some users reported that looking up from the data to the screen was uncomfortable, and they found it difficult to read the screen and see the data points at the same time.

Appendix 20: Data Panels' Text Shader

The text-mesh shader for Unity is, by default, both two sided, and ignoring of the z-index. The result of this is that it appears both from the front and the back, and it is drawn on top of everything, even if it is behind an opaque object in the world-space. For a user-controlled information panel, the text-mesh shader needed to be re-architected to account for only being one-sided and respecting its position in the z-index.

Appendix 21: Extended Conclusion for Experiment #2

Not needing to manage their own state (such as scale and position) meant users could focus on comprehending the data as a whole. They did however report that it was more difficult to

see the detail clearly, and leaning in closer was uncomfortable due to the HTC Vive's poor near-sight focusing abilities. This was especially problematic in areas where data was particularly dense.

An unexpected observation was that users were, at first, very hesitant to move through the virtual table. One user nearly lost their balance attempting to lean on the table. To promote movement through the table, and to prepare users for the non-solidity of the table, the table could be presented more like a hologram, complete with transparency. Once users had walked through the table once, however, they appeared much more comfortable moving through it in the future.

Users also liked having more control over the position of the data selecting controller. They typically found that they could select and navigate between data points faster and more reliably than before. The map being raised to approximately waist height (80cm above the floor) also helped users with more accurate selection, since they could make small hand movements from a relaxed arm position, which would allow for more comfortable continued use of the system for a longer time period.

Appendix 22: Description of "Natural" Controller Gesture System

Both tracked controllers are required for the pan, rotate, and scale gestures, which are activated by pulling in the triggers on both controllers at the same time. To pan, the user needs to move both controllers in the same direction, and the map will move the amount that the average position of the two controllers has moved on a per-frame basis. Rotation is handled by tracking the relative rotation of a vector perpendicular to a vector between the two controllers' positions on a per-frame basis. Scale is handled by the relative distance between the two controllers on a per-frame basis. These three virtual-reality gestures allow the controllers to be used to entirely manipulate an object along a two dimensional plane (in this case, the XZ plane).

Limitations on the scaling are put in place to prevent the user setting the scale to zero (or even below zero), or making the object so big it would be almost impossible to comprehend. Sensible scaling limits would need to be implemented on a per-manipulated object basis.

Given more time to experiment, the manipulation system could have benefitted from adding momentum to changes, so that if an object is scaled quickly, that scaling would continue for a brief period of time after the user depresses the triggers on the controllers. This would result in a physical interface more in step with how a user of an existing multi-touch interface system would expect a physical-based interaction of two controllers to interact in a virtual environment.

One issue that was discovered while implementing this feature was that without a sense of world-location users would get motion sickness moving the map. This was solved by making a clear horizon line visible, and adding a simple grid to the floor of the world. The ability to

ground the user in the world and let them clearly see that it's the map that's moving, not their head, vastly reduced the number of testers complaining of motion sickness.

Appendix 23: Issues Relating to the Development of a Custom Terrain System in Unity

An issue that was discovered early on in creating the new terrain system was that a mesh in Unity created in real time could not have more than 65,534 vertices. A 512 pixels^2 height map requires 262,144 vertices, and a 1024 pixels^2 height maps requires 1,048,576 vertices, so the terrain needed to be split into tiles of meshes, each falling under the 65,534 vertices limit. This involved creating sub-textures (sampled from the primary height map), and creating mini-terrains from the sub-height maps. These are then tiled together to make the appearance of one large terrain that could have many millions of vertices, made up of potentially dozens of sub-terrains.

Developing a mesh generation system was a lot more difficult than expected. It required building meshes from core-graphics principles: laying out vertices in grids with the height information extracted from the height map texture; setting their UV positions for the master diffuse texture; and creating the triangles from the vertices in the correct order. Developing this system required a lot of reading into how the Unity graphics subsystem handled laying out triangles from vertices, and how that affected their normal directions.

Appendix 24: User Testing Questionnaire

Introduction & Terms

Thank you for taking this user experience research survey. The information and feedback you provide will help software developers and designers build the next generation of software. The data being recorded for the survey cannot be linked back to you, and is stored in compliance with the Data Protection Act. The data will be controlled by the School of Computing and Mathematics at Plymouth University. It is being collected in order to make general recommendations based on how users use data visualisation software. The information recorded today will not be shared outside of the university, but averaged and collated data from multiple surveys will be used in research papers which can be distributed widely.

By signing below, you agree to these terms:

Signature:	Date:

Demographic Information

In this section we are collecting a small about of demographic information. All of it is optional, so if you're uncomfortable with any question, feel free not to answer it.

1. What is your age range? Please tick one:

Under 18	18 to 21	22 to 25	26 to 29	Over 30
I'd rather not disclose				

2. What gender do you identify with? Please tick one:

Male	Non-Binary	Female
I'd rather not disclose		

3. What hand is your primary hand (for example, for writing with)? Please tick one:

Left Handed	Ambidextrous	Right Handed
I'd rather not disclose		

Technical Baseline

In this section we want to find a bit about your history with technology.

4. Do you agree with this statement: I keep up to date with technology news.

Strongly Disagree	Disagree	Neither Agree, nor Disagree	Agree	Strongly Agree

5. Do you agree with this statement: I am technologically savvy.

Strongly Disagree	Disagree	Neither Agree, nor Disagree	Agree	Strongly Agree

6. On what frequency do you use mapping software (such as Google Maps, or Apple Maps) on a smartphone?

Never	Yearly	Monthly	Weekly	Daily

7. Have you used virtual reality before?

Yes	No
If you answered 'Yes', please note which devices you've used:	

Thank you for taking the survey. Tom will now take you through the demo.

Appendix 25: User Testing Process Sheet

Thanks for coming in, this user test will likely take about 10 minutes. Before we start, would you mind completing this questionnaire in order to get some demographic data and a technical baseline? Thanks.

GIVE THEM THE QUESTIONNAIRE, HAVE THEM COMPLETE IT.

Great. Before we jump into the demo, I'll explain quickly how you can control it, and the sorts of things you'll be doing.

You'll be looking at a dataset overlaid on a map of Plymouth and the sound. It is a dataset of road traffic collisions that occurred in Plymouth in 2012. The colour of the data points corresponds to its severity.

IF DOING VR DEMO:
You'll be trying out the virtual reality proof-of-concept demo. You'll use these controllers to interact with the virtual environment. You select a data point by pulling the trigger on the controller, and you can deselect it by doing the same again. You can extend or contract the halo on the controller by pressing the trackpad on the controller. If you pull both of the triggers, you can scale, rotate, or move the map, by moving the controllers as you would while looking at a picture on a touchscreen phone.

IF DOING NON-VR DEMO:
You'll be trying out this demo. You can interact with it using the mouse and keyboard. You can use the right mouse button to pan around the map, and the scroll wheel to zoom in and out. You can select a data point by clicking on it, and dismiss it by clicking on it (or it's information panel) again.

I'll give you some tasks. I'd ask that you wait until I'm done reading you the full task before you start doing it. You will be doing three tasks:

For the first task, I would like you to select a fatal road traffic collision, tell me when it happened, and then de-select it.

TIME TO COMPLETE:

MOVED WORLD:

ATTEMPTS TO IDENTIFY FATAL RTC:

CORRECT DATE & TIME:

For the second task, could you please select two data points by the ocean, and tell me the difference in the number of casualties between the two collisions. Once you're done you can de-select the data points.

TIME TO COMPLETE:

MOVED WORLD:

BY THE OCEAN:

SELECTED TWO:

CORRECT DIFFERENCE:

How many fatal accidents are there in the data set?

TIME TO COMPLETE:

MOVED WORLD:

RESPONSE (answer is three):

Appendix 26: Summarised User Testing Data

The following data is summarised from the raw data collected from users, with identifying information removed for the sake of anonymity.

26.1. Demographic Information

Demographic information was collected in order to discover whether any trends discovered could be attributed to specific attributes of users, instead of just as part of the software being tested.

User	Date	Gender	Handed	Tech News	Tech Savvy	Map Freq.	Used VR Before?	Headsets
1	06/05/2016	Male	Right	5	5	Monthly	TRUE	Rift, Vive
2	06/05/2016	Male	Right	4	4	Weekly	TRUE	Rift, Vive
3	06/05/2016	Male	Right	5	4	Monthly	FALSE	
4	06/05/2016	Male	Right	5	4	Weekly	TRUE	Rift
5	08/05/2016	Male	Right	4	5	Monthly	TRUE	Rift, Cardboard
6	08/05/2016	Female	Left	4	4	Daily	FALSE	
7	08/05/2016	Male	Right	3	4	Monthly	TRUE	Cardboard
8	09/05/2016	Male	Right	4	3	Weekly	FALSE	
9	09/05/2016	Male	Right	5	5	Weekly	TRUE	Vive, Cardboard
10	10/05/2016	Female	Right	5	5	Monthly	TRUE	Vive
11	12/05/2016	Male	Right	4	4	Monthly	TRUE	Rift, Vive
12	12/05/2016	Male	Left	2	3	Daily	FALSE	
13	12/05/2016	Male	Right	3	4	Weekly	TRUE	Rift
14	12/05/2016	Male	Left	5	4	Monthly	TRUE	Rift
15	12/05/2016	Male	Right	4	5	Weekly	FALSE	
16	14/05/2016	Male	Right	3	5	Monthly	TRUE	Vive
17	15/05/2016	Female	Right	4	4	Weekly	FALSE	

***Table 3**: The demographic information captured on the 17 users from the user tests.*

17 users were selected for user testing (see *Table 3*), 14 males and 3 females. 82% were right-handed, a little under the estimated world-average (~87% right handed).

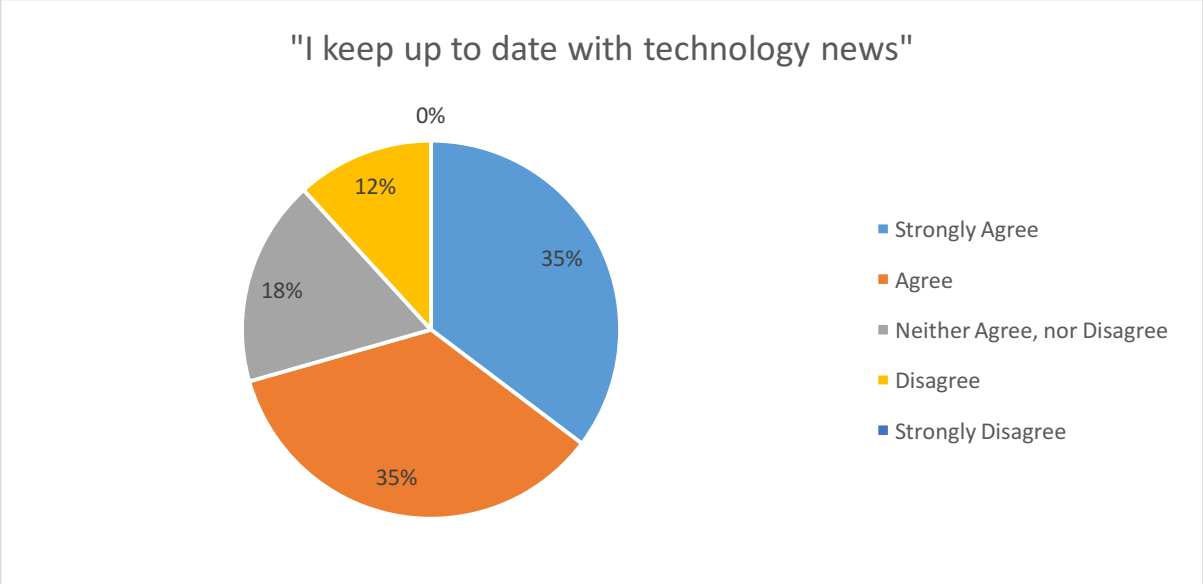

***Figure 15**: The vast majority (70%) of users reported that they did keep up to date with technology news.*

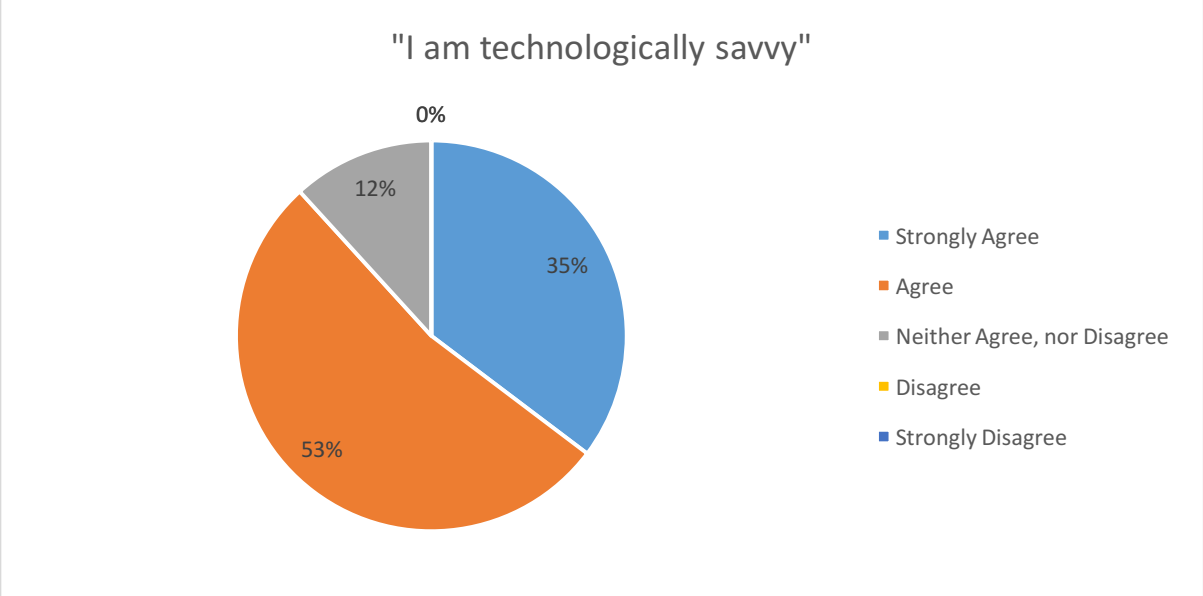

***Figure 16**: The vast majority (88%) of users reported to be technologically savvy.*

When asked whether they considered themselves tech-savvy, 88% said that they did consider themselves to be technologically savvy (*Figure 16*), and the remaining 12% said that they neither agreed, nor disagreed. 76% said they agreed that they "keep up to date with technology news" (*Figure 15*). 17% said they neither agreed nor disagreed, and 7% said they did not keep up to date with technology news. There was no significant correlation between how technological a user saw themselves and their speed and accuracy using the software proof-of-concepts.

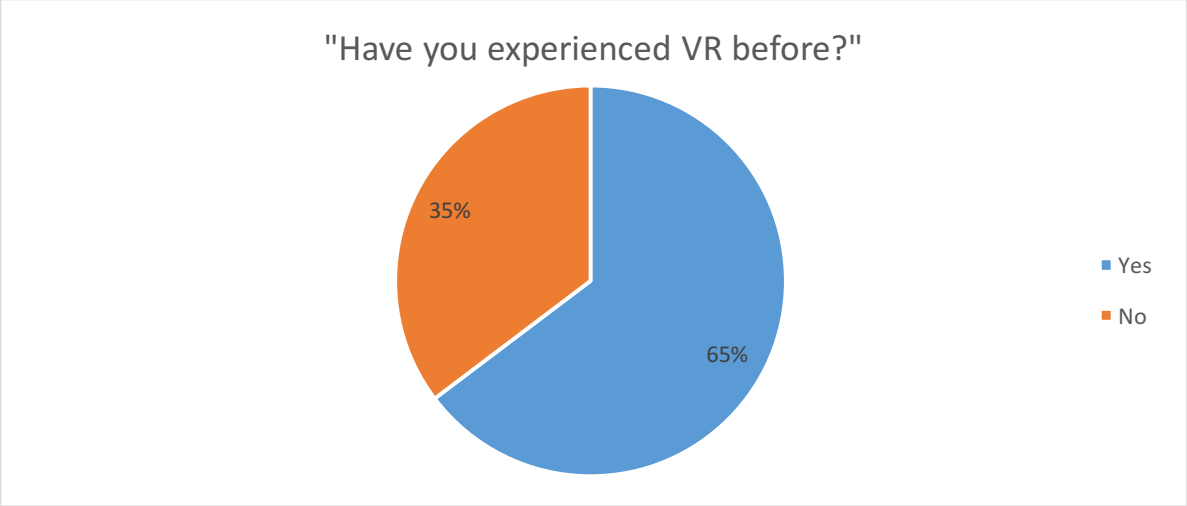

***Figure 17**: Most users had used virtual reality systems before, but a significant amount were trying it for the first time.*

11 of the 17 users (65%) had used a virtual reality system before (*Figure 17*), and 6 of those (35% of the total samples) had used the HTC Vive hardware. The most commonly experienced VR headset was the Oculus Rift, although the versions of the Rift (DK1, DK2, or CV1) were not recorded.

26.2. Movement in VR

It's important see how different users use the many degrees-of-freedom offered by the HTC Vive RSVR system. Some users will want to move a lot, and some will remain more stationary.

The dataset is too small to draw meaningful conclusions relating to whether users move their dominant hand's controller more (since there are only two left-handed users in the data set), as hypothesised. Right-handed users seem to move the right controller slightly more during the tasks, although it is difficult to be certain.

The mean amount of distance travelled by the various tracked controllers (*Figure 19*) can be used to determine how to optimise interfaces to maximise the length that a user can comfortably use the application, given that rate of fatigue is correlated with the distance a user travels.

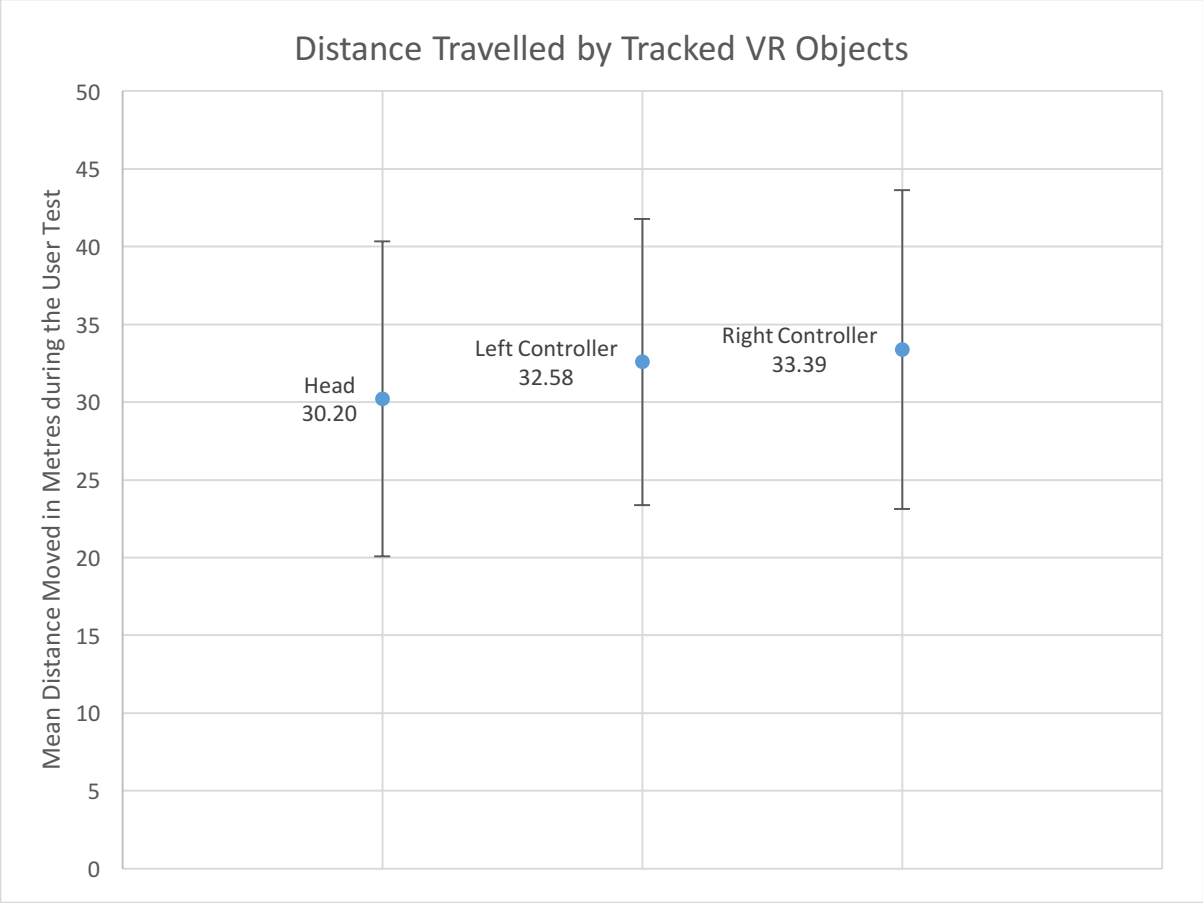

***Figure 18**: The mean movement of the head, left controller, and right controller over the length of the user test. Standard deviation is included as the error. What was surprising was that the head moved almost as much as the controllers, suggesting that users move their heads to see data as much as they move their hands to manipulate it.*

26.3. Manipulating the World

Another important part of building software that users can do productive work in is making the features and interactions discoverable and easy to use. The "Natural" Controller Gesture system, explained in *§7.3.1* and expanded upon in detail in *Appendix 22*, is the user's way to adjust their virtual environment to best enable them to work well.

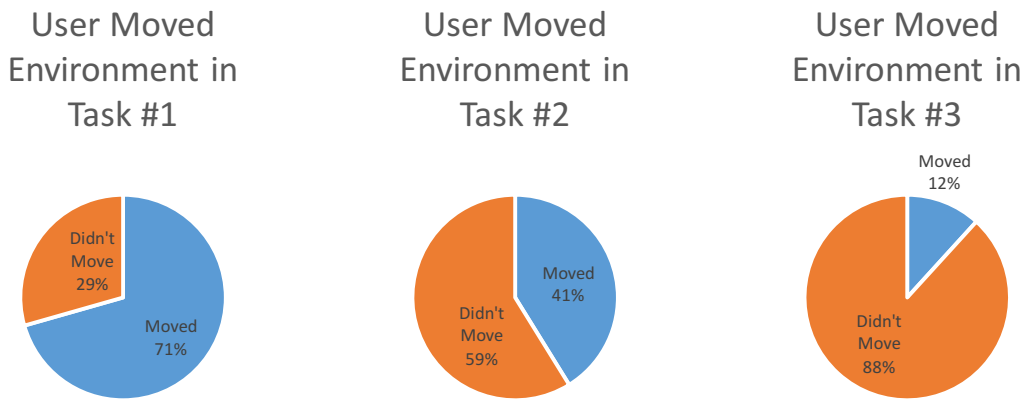

***Figure 19**: Whether the user used the "Natural" Controller Gesture system to modify the position, rotation, or scale of the map and the dataset.*

82% of users moved the environment in at least one task, but only 5% of users moved the environment in all three tasks (*Figure 20*) (*Table 4*).

User	Moved in Task #1	Moved in Task #2	Moved in Task #3	Moved At All?	Moved in All Tasks
1	TRUE	FALSE	FALSE	TRUE	FALSE
2	TRUE	TRUE	FALSE	TRUE	FALSE
3	FALSE	FALSE	FALSE	FALSE	FALSE
4	TRUE	TRUE	FALSE	TRUE	FALSE
5	FALSE	TRUE	FALSE	TRUE	FALSE
6	TRUE	FALSE	TRUE	TRUE	FALSE
7	TRUE	FALSE	FALSE	TRUE	FALSE
8	TRUE	TRUE	FALSE	TRUE	FALSE
9	TRUE	FALSE	FALSE	TRUE	FALSE
10	FALSE	TRUE	FALSE	TRUE	FALSE
11	TRUE	FALSE	FALSE	TRUE	FALSE
12	TRUE	TRUE	TRUE	TRUE	TRUE
13	TRUE	FALSE	FALSE	TRUE	FALSE
14	FALSE	FALSE	FALSE	FALSE	FALSE
15	TRUE	TRUE	FALSE	TRUE	FALSE
16	TRUE	FALSE	FALSE	TRUE	FALSE
17	FALSE	FALSE	FALSE	FALSE	FALSE

***Table 4**: Data on which users moved the environment in the three tasks, and analysis as to whether they moved the environment at all, or for each task.*

Appendix 27: Analysis of Reliability of User Testing

It is not reasonable to expect that every user test will be entirely representative, and this user test isn't. There are a number of problems with the sample chosen:

- The sample is overwhelmingly male. With a general population that is 51% female, the 17% representation in the sample is severely underrepresented. It is not hypothesised that gender has any role in one's ability to comprehend and act upon data sets either in, or out of, VR. Despite this, the huge skew against females is a cause for concern when developing general recommendations for RSVR paradigms that would affect women equally.
- It is not noted in the demographics, but the sample was taken in a computer science department of a university, in the United Kingdom. As a result of this, the users tested were predominantly 18-30, well educated, and technologically minded. The users also likely approach the application from a largely western mind-set, as it is the local culture.
- Most of the users had tried VR in the past. It would be fair to hypothesise that this is not the case for the general population, or even the data-scientist population. While this is explored in *Figure 13*, it would have been more representative to have more users who had never tried VR before. It could be argued, however, that if the findings of the project are intended to develop paradigms for every-day use of RSVR, it could be beneficial to overcompensate with users who have experience with VR, since it could be more representative of a future population the recommendations could benefit.

While no qualitative data was collected in the final user test, feedback was collected informally throughout the experimentation process. The author was careful to take feedback from first-time RSVR users with a grain of salt, since the overwhelming "wow-effect" of the technology could cloud their judgement of how good the software interactions are.

Appendix 28: Software Scope Objectives

The three requirements in the initial scope were as follows:

1. *"The proposed core interface system will allow users to load a dataset into software to be visualised"*

The final software solution loads a dataset automatically from a defined CSV file. The software then visualises the dataset using the location information discovered from the CSV dataset, as outlined in *§7.1.1*.

The original intention of this scope item was to enable the user to hot-swap generic datasets. However, as the project moved from a generic data viewer to a more specialised geovisualisation application, the requirement to load and unload data waned.

2. *"The proposed core interface system will allow users to view the dataset by moving around the RSVR system."*

The software solution allows users to view the loaded dataset from any angle they can physically move to.

3. *"The proposed core interface system will allow users to scale themselves to be able to adapt to small variances in data, as well as large datasets that could span tens of kilometres."*

Appendix 29: Agile in a One-Person Team

Agile is a good methodology for rapidly creating products, and if the project had taken place with a team of designers and developers, I think it would have been ideal as a process. However, as a single-person project, a lot of the core fundamentals of agile development were a hindrance instead of a blessing. If I were to do the project again, I would move further from the textbook agile process, bringing only the story-based progress measurement and the continual re-evaluation of the direction the project is moving in. These two aspects of agile worked well, even with a one-person team.

Appendix 30: Raw Data

For demographic data, be sure to refer back to *Table 3*, in *Appendix 26*. The following raw data and its users' anonymous identifying numbers refer back to *Table 3*.

User	Time (sec)	Moved World	Attempts Req.	Correct Date/Time
1	6	TRUE	1	TRUE
2	8	TRUE	1	TRUE
3	16	FALSE	1	TRUE
4	5	TRUE	1	TRUE
5	7	FALSE	1	TRUE
6	12	TRUE	2	TRUE
7	5	TRUE	1	TRUE
8	10	TRUE	2	TRUE
9	5	TRUE	1	TRUE
10	6	FALSE	1	TRUE
11	4	TRUE	1	TRUE
12	21	TRUE	3	TRUE
13	14	TRUE	2	TRUE
14	7	FALSE	1	TRUE
15	4	TRUE	1	TRUE
16	8	TRUE	1	TRUE
17	9	FALSE	1	TRUE

***Table 5**: Raw Data on Task #1*

User	Time (sec)	Moved World	By Ocean?	Selected 2?	Correct Difference?
1	17	FALSE	TRUE	TRUE	TRUE
2	16	TRUE	TRUE	TRUE	TRUE
3	22	FALSE	TRUE	TRUE	TRUE
4	19	TRUE	TRUE	TRUE	TRUE
5	20	TRUE	TRUE	TRUE	TRUE
6	24	FALSE	TRUE	TRUE	TRUE
7	15	FALSE	FALSE	TRUE	TRUE
8	16	TRUE	TRUE	TRUE	TRUE
9	14	FALSE	TRUE	TRUE	TRUE
10	12	TRUE	TRUE	TRUE	TRUE
11	16	FALSE	TRUE	TRUE	TRUE
12	20	TRUE	TRUE	TRUE	FALSE
13	14	FALSE	TRUE	TRUE	TRUE
14	15	FALSE	TRUE	TRUE	TRUE
15	9	TRUE	TRUE	TRUE	TRUE
16	14	FALSE	FALSE	TRUE	TRUE
17	12	FALSE	TRUE	TRUE	TRUE

***Table 6**: Raw Data on Task #2*

User	Time (sec)	Moved World	Correct Answer?
1	7	FALSE	TRUE
2	6	FALSE	TRUE
3	11	FALSE	TRUE
4	4	FALSE	TRUE
5	5	FALSE	TRUE
6	4	TRUE	TRUE

7	7	FALSE	TRUE
8	6	FALSE	TRUE
9	5	FALSE	TRUE
10	9	FALSE	TRUE
11	10	FALSE	TRUE
12	11	TRUE	TRUE
13	7	FALSE	TRUE
14	8	FALSE	TRUE
15	5	FALSE	TRUE
16	10	FALSE	TRUE
17	8	FALSE	TRUE

***Table 7**: Raw Data on Task #3*

User	Head (m)	Left Con. (m)	Right Con. (m)
1			
2	14.82	24.95	19.74
3	34.2	40.33	28.71
4			
5	24.43	30.12	39.73
6			
7			
8	31.92	26.6	45.32
9			
10	41.01	31.61	37.48
11	24.75	17.58	22.84
12	51.3	50.52	43.89
13			
14	25.43	31.39	41.96
15	26.64	40.04	36.12
16			
17	27.54	32.65	18.09

***Table 8**: Raw Data on User Movement (only VR users)*