UNIVERSITY OF ABERTAY

HONOURS DISSERTATION

Adaptive User Interfaces For Virtual Reality

Author:

James WOOD

Supervisor:

Dr. Paul ROBERTSON

March 28, 2017

CONTENTS

List of Figures

List of Tables

1 ABSTRACT

Virtual reality allows users to interact with three-dimesional environments and interface in an intuitive way, looking around with the head mounted display and reaching out with wireless tracked controllers. Although VR interfaces often take advantage of the new space and depth afforded by the stereoscopic 3D inherent in a head mounted display, they are generally still static like the 2D interfaces prevalent on other devices.

This project expands on research made in the area of adaptive user interfaces which focused on 2D interaction, applying techniques in the 3D, motion controlled environment of VR. Adaptive user interfaces aim to improve usability with dynamic elements and input analysis. The two techniques focused on in this project are error analysis and adaptive scaling. To implement an application for evaluating these techniques C++ and Unreal Engine 4 were used.

On of the most common and difficult to use interaction models in virtual reality is the laser pointer. Interacting with far away elements can be cumbersome using a pointer and is an area in which adaptive techniques can easily be applied. As such the application used for testing during this project has the user perform simple selection and deselection interactions on a set of objects using a laser pointer.

To evaluate the benefits of the adaptive techniques applied in the application, both quantitative questionnaire data and qualitative data have been gathered and analysed. To measure the usability of the adaptive techniques the process used a static control interface. Combing the two sets of data and comparing gives a clear representation of the effect the adaptive techniques have on usability.

Keywords: Virtual Reality, User-Interfaces, 3D Interfaces, Adaptive Interfaces, Interface Plasticity

2 INTRODUCTION

2.1 THE FALL AND RISE OF VIRTUAL REALITY

Virtual reality (VR) head mounted displays first started appearing in the commercial market in the 1990s with Nintendo's Virtual Boy as the face of what was sold as a new era in entertainment technology. The virtual boy and its ilk failed miserably causing sever motion sickness and utilising displays far worse than those available for the standard consoles at the time. Despite this terrible start, two decades later VR has begun to secure its place in mainstream entertainment especially in the world of video games, although Nintendo is nowhere to be seen.

Sony's PlayStation VR headset was more successful than predicted (Seeto 2016) and the fears surrounding the motion sickness that doomed the previous wave of VR are dissipating (Pino 2016). Alongside Sony's offering are HTC's Vive and Facebook's Oculus not mentioning the vast number of headsets designed for smartphones. These three headsets all provide similar experiences: high resolution displays for each eye providing stereoscopic depth, six degrees of freedom head tracking and hand tracking using wireless controllers.

There have been some stand out applications and games from these early years of the new VR platforms. Job Simulator somehow turns the mundane tasks of everyday life into fun, captivating play and would not be the same experience without the head and controller tracking that comes with the HTC Vive. Tilt Brush was the first breakthrough success on the creation front with users creating thousands of sculptures using the signature 'light ribbons'. It showed that the intuitive 3D of VR could be used to great effect in productivity applications.

Even before the first release of the main VR platforms as consumer products the major Game Engine creators, Unity and Unreal Engine, had begun implementing VR support not just for applications but for the editor used to create applications. This allows not just VR developers but all developers to utilise the precision that comes with the spacial awareness to edit 3D environments with ease.

(a) Job Simulator (Owlchemy Labs 2016)

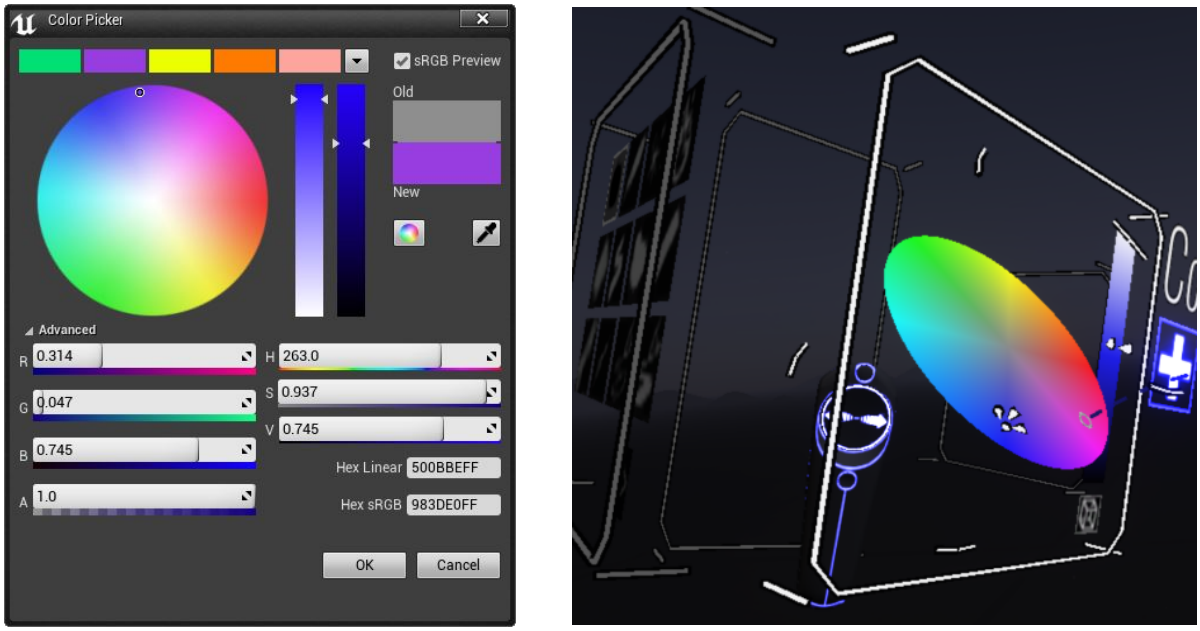
(b) Tilt Brush (Google 2016)

Figure 1: Examples of Virtual Reality applications

2.2 THE FUTURE OF IMMERSIVE TECHNOLOGIES

The features of the forerunning virtual reality headsets combine to create a sense of immersion or 'presence' rarely achieved in traditional 2D applications and games. Even though this new wave of VR has only been around for a few years a new technology is rapidly approaching. On the horizon is the next step in immersive technology, augmented reality (AR). This will allow a wireless experience and build upon the work in mobile virtual reality by Google and Samsung. Microsoft's Hololens provides a glimpse at this future, one of seamless multi-user interaction overlaid onto reality.

Something often associated with VR is the isolation wearing a headset implies. AR reverses this by allowing the user to still see and interact with the real world while experiencing an immersive experience. The Hololens, even in its primitive first form, demonstrates how this can introduce new forms of interaction not possible in any other way. The ability for two people to interact and collaborate with the same interface while still seeing their surroundings with no constraint on location is revolutionary.



(a) 2D picker (UnrealEngine4 2012)

(b) 3D colour picker (Tilt Brush 2016)

Figure 2: Comparison between 2D and 3D interface

2.3 Interfaces in 3D

Although everything studied and evaluated in this project can apply to any 3D interface, the testing and implementation focus on virtual reality as a platform, specifically the HTC Vive. Current virtual reality interfaces focus on interaction using the controller itself which constrains elements to being within arms reach.

A common solution is a 3D laser pointer used to select and manipulate distant elements. As in the real world using a laser pointer at a distance can be imprecise and slow (Malaika 2015) when under pressure. It is however an intuitive and unobtrusive mechanic and easily translates from the traditional desktop input of a mouse cursor.

This project evaluates how the usability of 3D interfaces in VR, specifically those utilising a 3D pointer system, can benefit from being dynamic and adaptive. It references concepts and techniques such as error analysis, adaptive scaling and user modelling explored previously by other researchers in 2D and 3D. Figure 2 shows a comparison between a traditional 2D interface and a 3D virtual reality interface.

2.4 RESEARCH QUESTION

The project focuses on a single point: *"How can the usability of virtual reality interfaces benefit from adaptive techniques?"*. In order to begin implementing the application and prepare for evaluation this research question was separated into several aims and objectives as follows:

Aims:

- Apply adaptive techniques to a static virtual reality user interface.
- Gather quantitative data through user testing and error analysis.
- Gather qualitative data using questionnaires.
- Evaluate the overall effect of adaptive techniques on the usability of a virtual reality interface using the data gathered.

Objectives:

- Research adaptive user interface techniques for 2D and 3D interfaces.
- Implement a 3D user interface for use in virtual reality.
- Employ the researched techniques for assessment.
- Create a system for error analysis to gather qualitative evaluation data.
- Have users test the iterations of the interface and feedback on their usability.
- Compare the various techniques using the qualitative and quantitative data gathered to evaluate their effectiveness.
- Using the comparisons evaluate the overall impact of adaptive user interfaces on usability in virtual reality.

3 LITERATURE REVIEW

3.1 ADAPTIVE USER INTERFACES

Dynamic user interfaces fall under a few categories in the area of research sometimes referred to as 'interface plasticity'. A survey of plasticity in 3D user interfaces' (Jeremy 2014) gives an overview of the different types of dynamic user interface classifying them using two parameters, the adaptation time and the controller. Within these parameters an adaptive user interface is defined as being adapted by the system at run-time. This project focuses on implementing systems that fall under this classification of adaptation.

Many researchers have explored various ways to assist the user in digital environments since the early days of personal computing (Morgan 1998). One strategy known as intelligent user interfaces, seeks to actively learn how a user interacts with an interface and suggest or highlight relevant information accordingly. Although this can be effective, it takes time to model the user in a meaningful way and can seem obstructive as demonstrated by Microsoft's Clippy (Meyer 2015).

Another approach, the one this project focuses on, is to adapt the interface itself in direct response to the user's actions. One of these techniques covered by 'Modeling error-based Adaptive User Interfaces' (Kathik 2011) monitors the user's failures while navigating and interacting with an application to dynamically interpret their desired interaction. 'Adaptive Hypermedia and Adaptive Web-Based Systems' (Peter 2000) discusses the benefits of adapting a 3D environment as the user interacts with it to assist their actions.

Most user interfaces in our digital lives are static with adaptive interfaces reserved for improving accessibility for atypical users or use-cases. VR provides a new platform for innovation and does not have a direct counterpart to the universal language of 2D interfaces' windows and pointers. This uncharted territory provides an opportunity to experiment with dynamic 3D interfaces in new ways.

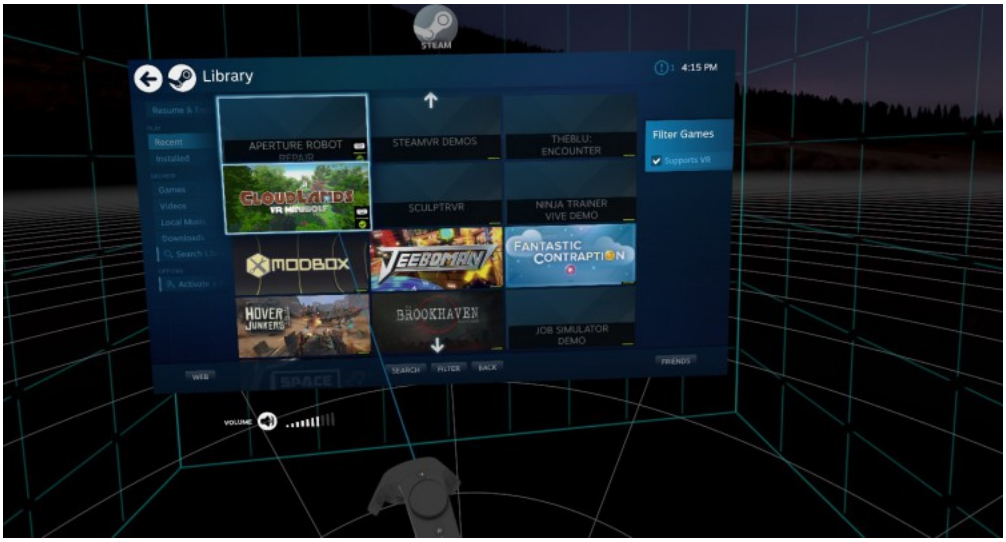


Figure 3: Pointer based interaction (Steam VR 2016)

3.2 VIRTUAL REALITY USER INTERFACES

When considering the current landscape of virtual reality applications and games there are a large number using motion controllers as opposed to traditional input methods such as console controllers and keyboard/mouse. The HTC Vive's lighthouse tracking system provides accurate motion control walking around a large space up to 15 ft square. The Oculus' and PSVR's use of optical tracking lends itself to a seated experience but still allows full 360 degrees of movement.

Considering that most content on VR platforms is not exclusive to a single device experiences focus on interaction at arms length to accommodate for seated users. Interactions often consist of selecting 3D objects or interface elements by moving a controller to their position ad pressing a button. Any interaction beyond arms length generally utilises a 3D pointer system as seen in the Steam VR interface (Figure 3) used to enter applications for the HTC Vive.

These pointer interfaces become more cumbersome to use the further away, smaller and more complex the interface becomes. Despite this there are few interfaces using dynamic or adaptive elements to improve their usability. By simply assisting the user in their interaction great improvements could be made to bring VR closer to replacing desktop PCs and traditional 2D displays for everyday tasks like word processing and file management.

3.3 USABILITY IN VIRTUAL REALITY

The companies and developers at the forefront of the new wave of virtual reality hardware and software have faced many challenges in comfort and usability. Issues with nausea and motion sickness plagued early iterations of head mounted displays. The addition of positional tracking and a greater understanding of significant design choices like locomotion have overcome most of these growing pains.

Now the focus of research is around usability of interfaces in VR. Valve, a driving force in desktop VR, presented some factors of usability at the Game Developer's Conference. When discussing interactions outside the user's area of reach they cited Fitt's law (Malaika 2015). This law has been extended from one dimensional interfaces into 2D interfaces and now into 3D and states that the speed at which a user acquires a target is exponentially proportionate to the size of the target at a given distance (Atsuo 2001). When using a pointer based system, small interface elements at a distance away become very slow to interact with.

Another notable aspect of ease-of-use regards the placement of interface elements around the user. Due to the 360 degree nature of a 3D interface if the user is constantly required to turn around in order to perform actions it will become fatiguing. Owlchemy labs, developers of Job Simulator and strong proponents of diegetic interfaces build scenarios in such a way that users do not have to turn frequently and generally have direct access to everything they may need at a given time (Schwartz 2015).

3.4 POSSIBLE DISADVANTAGES OF ADAPTIVITY

Although adaptive user interfaces aim to assist a user, they could also be obstructive and counter intuitive when misused. Users are accustomed to interfaces being static and to having full and direct control over the elements presented to them. Adaptive techniques can break some of these expectations by manipulating the elements displayed and interpreting actions made by the user. This extra layer of system adaptation could in some situations work against the user.

4 METHODOLOGY

4.1 APPLICATION

4.1.1 OVERVIEW AND DESIGN

Before being able to apply the two adaptive techniques, dynamic scaling and error analysis, an interface needed to be implemented. The design of the interface and the application as a whole was designed with user testing and data gathering in mind. In order to better understand what the interface would need to provide, several points were devised to outline the core features and constraints:

- The interface must feature a minimal number of clear interactions for easy learning.
- The interface must use a laser pointer to interact with elements outwith arms reach.
- A pressurised environment should be used to encourage users to interact with the interface.

Each of these points was taken into consideration and a simple virtual reality application was developed. The interactions considered were those common to many applications but a drag and drop interface was decided upon. This would be intuitive for the users who would only have a few minutes to use the application. Many pieces of existing virtual reality software show that moving and throwing objects in virtual reality is a gratifying action. Having decided upon the interaction the next step was to select objects to be interacted with.

To maintain the simplicity of the application and to try and avoid users becoming wrapped up in the details of the objects they were interacting with a simple ball was chosen. Initially a more complex object was implemented but as the focus needed to be on interacting and not the objects being interacted with the coloured ball leant itself to the testing. A ball also has associations with grabbing and throwing which reinforced the core interaction.

The choice to use a laser pointer was made early on as there is inherent difficulty in the task of selecting something in 3D space using a pointer. The use of a pressurised environment was

executed in the form a timed and scored interface. At first a kind of tutorial in which the user would be prompted to perform different actions was considered. The simple timer system was chosen instead because it did had no reliance on the speed at which the user reads or the interpretation of instructions. By having a single task repeated as many times as possible the user is forced to be less cautious with their interactions, providing more varied data.

The final design of the application took into account the outlined points and consisted of a simple space with a table to focus the user's gaze. On this table balls of two colours appear and roll toward the participant. A timer is visible on the table and is counting down to zero, some text instructs the user to place the balls into a basket of corresponding colour next to the table. Each time the user successfully selects and places a ball a score readout above the basket increases.

4.1.2 Implementation and Iteration

The initial design was implemented (as seen in Figure 4) in Unreal Engine 4 using C++ to create the core interaction system and Blueprints (visual scripting) to create the gameplay. By separating the code of the gameplay from that of the interaction it allowed easy iteration and tweaking of the surface features while having a strong, performant foundation in C++.

Once the implementation was finished an early user test was performed which revealed several issues with the design. Initially two controllers, on in each hand, were used to interact. Both had laser pointers but only one would be used in the dominant hand the other would occasionally obstruct the user's actions. The first implementation also asked the user to place blue balls into one basket and red into another, this meant as the balls appeared the user was constantly interacting and had no time to break.

Although the pressure on the user was intended it quickly became tiring for them to interact and the decision was made to have only one basket so that users would interact with only one colour and need to avoid the other. This would also give more data points for when the user did select a non basket coloured ball the system could tell it was a failed action.

Figure 4: Early Iteration

One change made before the final round of user testing expanded upon the interface itself. The applications in a real piece of software for selecting moving objects like the balls rolling toward the user are difficult to imagine. To improve the evaluation and provide a more realistic scenario the interface switches half way through the application's timer from the rolling balls to a grid of static balls suspended away from the user. This also tests the ability to select tightly packed objects something the moving interface of the rolling balls did not do effectively.

4.2 Adaptive Techniques

4.2.1 Overview

The research done into adaptive user interfaces and the techniques behind them focused mainly on implementations in 2D scenarios. To apply all of these techniques to 3D would be outside the scope of this project and have made it very difficult to complete the user testing. As such two techniques were chosen and altered for virtual reality. The first is error analysis, with the application requiring

only one button for interaction the room for error is small and so easily tracked. The concept of error analysis is to detect when a user has incorrectly pressed a button and somehow react to it.

The second technique chosen was dynamic scaling although in a different form to that found in most of the previous research done on adaptive techniques. Generally the scaling of user interface elements would be done in response to an error, improving a user's ability to select it. By relating the scaling of every element to the predicted selection of the user it can constantly accommodate and assist the interactions with the interface, scaling up areas of focus.

4.2.2 Error Analysis

To understand how to know when a user has made an error the underlying interaction system in C++ has to be explained. When designing the interaction system the input is bound to a set of functions. This means that when the user presses the trigger in an attempt to select an element it will run through a series of steps to complete the action:

- The user presses the trigger.
- The system runs the grab function.
- A raycast is sent along the laser pointer.
- If the raycast intersects any number of elements the first is selected.
- If the raycast intersects no elements, nothing happens.

It can be assumed to a certain probability that if the final step is true and no elements were found, the user made an error. The user could simply have pressed the trigger with no intention to select something but this is something that cannot easily be predicted. But in the case that the user was trying to make a selection and failed error analysis can allow the system to make an educated guess at what the desired element was and take action on it.

Although this assisted selection would be easily accomplished in 2D by simply choosing the closest element to the cursor, in 3D with a continuous 'beam' cursor going into infinite it is more complex to solve. The implementation uses the shortest distance formula to find which element

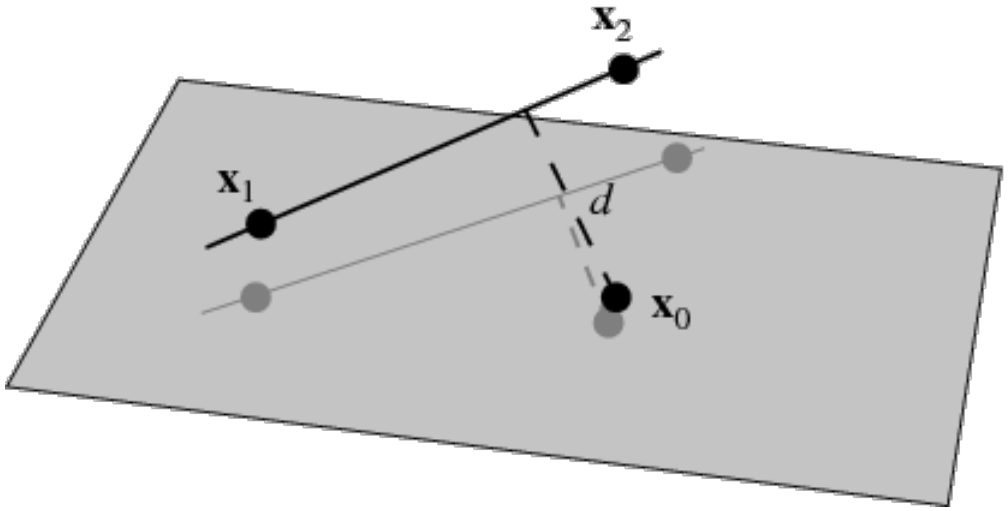


Figure 5: Visualisation of the shortest distance between a vector and a point

along the line of the laser pointer is the closest taking the magnitude of the vector forming a right angle to the direction.

By using a threshold distance beyond which elements are ignored the instant the user makes an error the system can search for any that are not overlapping but close enough to select and find the nearest. The basic form of this formula is as follows:

$$d = |(\mathbf{a} - \mathbf{p}) - ((\mathbf{a} - \mathbf{p}) \cdot \mathbf{n})\mathbf{n}|$$

In which d is the shortest distance between point p and the vector with position a and direction n. A visualisation of the formula can be seen in Figure 5 below. This implementation immediately proved to make selecting moving elements and densely packed elements faster and less cumbersome during basic testing. It could break down when a user is selecting objects very far away through a grouping of close objects in which case the formula could be modified to take into account distance of the perpendicular point along the laser.

4.2.3 Dynamic Scaling

The second technique adapts the interface visually in an attempt to improve the usability of smaller or tightly packed elements such as the grid of balls presented in the latter half of the application.

By using the shortest distance formula again to determine how far away an element is from the user's laser it can be scaled by some amount to create the effect of it becoming larger as the user goes to select it. The effectiveness of this technique relies on the theory laid out in Fitt's Law stating that the time to select a user interface element is proportionate to its size and the distance from it. By scaling the elements as the distance to it decreases the speed of the selection should increase.

Initially a more complex algorithm was used to scale the elements exponentially and with a slight downsizing at the start. This was intended to make a kind of focus bubble inside which elements would be larger and on the edge of which elements would be smaller. This concept made selection confusing especially when approaching a group of elements from far away as they would all shrink on approach before getting larger at varying rates.

Something that became apparent during the implementation of the adaptive scaling was the importance of consistency. Early testing made it seem that if the user expects something to happen to the interface dynamically that becomes something they will compensate for in the interaction with the interface. As soon as something breaks the consensus it is no longer reliable and could hinder usability.

The scaling of the object was applied to the mesh itself but not the collider, this would cause issues where the mesh would begin clipping into nearby static, unelectable meshes and would sometimes seem like a property of the element and not the interface in general. By simply adding a kind of highlight replica of the mesh with transparency and scaling this instead, each element could be seen at its native scale at all times yet the dynamic scaling could still be applied. This highlight mesh was also used for the static interface to show when an object was in range to be selected by becoming visible.

4.2.4 FURTHER CONSIDERATIONS

When implementing these adaptive techniques there were several other approaches considered. Stabilisation of the laser pointer could be used to snap to nearby elements like a pre-emptive error

analysis so that if the user would have slightly missed an element the system snaps the laser to it before they even select it. This was not used as the constant change of direction and lack of fine control over the laser could be disorientating and nauseating.

Another way to adapt the interface would be by modelling the user and adjusting the environment to them, this could be done in a number of ways. For example taking the user's height and scaling the entire interface to be proportionate to them. As the focus of the research and implementation was on an experience that could be accessed and used in only a few minutes the calibration required for this technique went against the project design.

Other than the input of the laser pointer and the button for selection the direction the user is facing could perhaps be used to improve the error analysis. This approach would take not only the closest element to the laser into account but also the closest element to the gaze of the user. As the user testing would already require the control test alongside the two techniques adding a third test for this iteration on error analysis did not seem beneficial.

4.3 EVALUATION

4.3.1 OVERVIEW

To evaluate the effectiveness of the two adaptive interface techniques a single virtual reality interface was constructed as described in 4.1.2, the basic version of this was used as the static control. Alongside this interface, which users would interact with for a minute under a timer, the same interface but with the adaptive techniques applied individually. This would give the data to compare and evaluate their impact on usability.

4.3.2 DATA GATHERING

By gathering data from the application in the background the performance of the user could be analysed and compared for the static control and each technique. Alongside the questionnaire

data this would give a solid understanding of how the user perceived the usability and how their interaction with the interface was actually affected.

The data gathering system was a simple messaging model that wrote out an xml formatted file which could then be read back into the Unreal Engine Editor and played back in 3D akin to a video capture. This proved to be useful when looking for errors in the development as well as for analysing a user's performance.

A single object, the data gatherer, writes to a file at a specified tick while the application is running. Every tick it stores the name, position and rotation of every object it is currently tracking. In order for an object to be tracked it must have a specific data generator component used to communicate with the data gatherer. When initialising this component the data gatherer adds the object's name to its list to keep track of it. The name can then be used to request the required information.

Although this data alone is enough to reconstruct a user's interactions with the interface visually it does not keep track of the results of their interactions. To do this a second event layer was added the communication between data generator components and the data gatherer. When the component creates an event the data gatherer stores it at the tick during which it occurred along with the name of the object that generated it and some description. This allows every error, success and other non-visual interactions to be store and replayed easily.

At first the data gathering system would output large amounts of data every frame causing performance loss and discomfort when using the application. To account for this a simple distance threshold was added so that each tracked object's information is only updated when it has moved a certain amount. When replaying data in the editor a simple interpolation can smooth out this loss of data.

4.3.3 QUESTIONNAIRE

To allow the users to feedback on the application and provide a counterpoint to the data gathered by the application which may not fully represent the user experience, a series of questions was devised and given to the user between the testing of each technique and the control (See Appendix A). The questions aimed to provide a variety of both negative and positive feedback while being vague enough to allow users to focus on the experience and not be caught up in their interpretation of each question.

The final form of the questions is an iteration on the System Usability Scale which is widely used as a tool for measuring usability of a system. The decision to use the term system instead of interface would allow the user to consider not just the physical interface but the interaction with the interface as well. The scale uses a five option system ranging from 'strongly agree' to 'strongly disagree'. This range gives enough room for comparison without overwhelming the tester with choices.

As in the standard Scale 10 questions were asked although slightly adapted from the ones given. The main addition was the final question regarding comfort. A vital component of usability in virtual reality specifically is user comfort. As there is a tendency for user's to associate their comfort with the experience itself by using this question to weigh the answers of the testers any especially bad experiences and the answers associated could be offset.

When user testing the order of the tests for the static control interface and two adaptive techniques was randomised. The user would enter the room, be briefed and then asked to complete the consent form. Once they were ready they were placed in the headset and left to interact with the minute long timed interface. When the timer expired the headset was removed and they completed the questionnaire. This was repeated until each iteration of the interface had been tested in a random order. To ensure that each user tested each interface in each order a balanced number of times a simple list of the interface iteration to be tested was devised and followed rather than letting the system randomise the techniques applied.

5 Results

5.1 Qualitative Data

5.2 Quantitative Data

5.3 Comparison

6 Discussion

7 Conclusion

8 APPENDICES

8.1 APPENDIX A

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

9 REFERENCES

Anastasios, K. 2012. *When You Shouldn't Use Fitts's Law To Measure User Experience* Smashing.

Atsuo, M. 2001. *Extending Fitts' law to a three-dimensional pointing task* Japan. Elsevier.

Brusilovsky, P. 2000. *Adaptive Hypermedia and Adaptive Web-Based Systems* Italy. Springer.

Bye, K. 2016. *Five Universal Tasks of 3D User Interfaces with Doug Bowman* Voices of VR. Available at: http://voicesofvr.com/five-universal-tasks-of-3d-user-interfaces-with-doug-bowman/ [Accessed November 2016]

Jen, H. 2016. *Designing for VR: Applying Usability Heuristics to Virtual Reality* Omobono.

Jeremy, L. 2014. *A survey of plasticity in 3D user interfaces* Minneapolis. IEEE.

Karthik, R. 2011. *Modeling error-based Adaptive User Interfaces* Iowa State University.

Malaika, Y. 2015. *Interaction Design in VR: Valve's Lessons* Available from: https://www.youtube.com/watch?v=_vQo0ApkAtI [Accessed October 2016]

Merki, R. 2016. *How VR Analytics Makes Your Product Better* Available from: http://cognitivevr.co/blog/how-vr-analytics-makes-your-product-better/ [Accessed October 2016]

Meyer, R. 2015 *Even Early Focus Groups Hated Clippy* Available from: http://www.theatlantic.com/technology/archive/2015/06/clippy-the-microsoft-office-assistant-is-the-patriarchys-fault/396653/ [Accessed October 2016]

Morgan, K. 1998. *Intelligent User Interfaces: An Introduction* San Fransisco. RUIU

Northway, C. 2016. *Menus Suck* VRDC. Available at: http://www.gdcvault.com/play/1023668/Menus [Accessed November 2016]

Pino, N. 2016 *HTC Vive review* Available from: http://www.techradar.com/reviews/

wearables/htc-vive-1286775/review/3 [Accessed October 2016]

Robert, Z. *Look-That-There: Exploiting Gaze in Virtual Reality Interactions* Providence. Brown University.

Schwartz, A. 2015. *Being There: Designing Standing VR Experiences with Tracked Controllers* Available from: https://www.youtube.com/watch?v=hjc7AJwZ4DI [Accessed October 2016]

Seeto, D. 2016 *Gamestop: PSVR Exceeding Expectations As More Units Are Received For Holiday Season* Online. http://attackofthefanboy.com/news/gamestop-playstation-vr-exceeding-expectations-units-received-holiday-season/ [Accessed October 2016]

Tilt Brush. 2016. [software]. HTC Vive. Google.

Unreal Engine 4. 2012. [software]. Windows. Epic Games.

10 BIBLIOGRAPHY

Alger, M. 2015. *VR Interface Design Manifesto* Available from: https://www.youtube.com/watch?v=n3b8hZ5NV2E [Accessed October 2016]

Casey, H. 2015. *Designing For Virtual Reality* Online. UsTwo. [Accessed October 2016]

Julie, J. 2011. *Human-Computer Interaction* 2011. Springer.

Max, G. *The UX of VR* Available from: http://www.uxofvr.com/ [Accessed October 2016]

Nasoz, F. 2004. *Adaptive Intelligent User Interfaces with Emotion Recognition* Orlando State University.

Rhino, K. 2016. *Building 3-Dimensional UI for VR* Available from: http://www.gdcvault.com/play/1023652/Building-3-Dimensional-UI-for-VR [Accessed October 2016]

Shaw, C. *Pain and Fatigue in Desktop VR: Initial Results* Canada. University of Regina.

Stanney, K. 1998. *Human Factors Issues in Virtual Environments: A Review of the Literature* Orlando. MIT.

Sutcliffe, A.G. *A Usability Evaluation Method for Virtual Reality User Interfaces* Manchester. UMIST.

Viano, G. 2000. *Adaptive User Interface for Process Control based on Multi-Agent approach* Italy. ACM.