Design and Development of a Virtual Reality Application to Introduce Gesture-Based Interaction

Johan Källberg

2016

Master's Thesis

Department of Design Sciences
Lund University

Abstract

This thesis provides insight into the development of Virtual Reality(VR) applications with the purpose of introducing users to a new gesture interaction platform. Gesture-based interaction and VR are two rapidly evolving technologies with a great potential to complement each other.

The gestures used are based on a cognitive model of holding a sphere in your hands and tapping on its surface. By using the hand tracker Leap Motion and the Head Mounted Display Oculus Rift the sphere shaped controller is placed in the user's virtual hands. The thesis is a collaboration with Erghis Technologies AB who developed the concept of the sphere and software to track the gestures.

Gesture based interfaces are called natural user interfaces. But a natural experience should be easy to understand and meet the users' expectation to interactions. To meet this design challenge the interface was broken down into the actions the user can perform and making sure that they were conveyed to the user and reinforced with feedback. The flow between user action and system feedback was fine tuned for every gesture to improve the feeling of the interaction. When using the system the users are first shown the gestures on a video played in the virtual environment. After the short introduction they can form an understanding about the gestures by interacting with the system.

For the interaction loop to flow the gestures needs to be reliably tracked. The gestures are easier to track when the hands are easy for the camera to see. This means that gesture interfaces designed for the users to position their hands in a way that can easily bee seen by the tracking camera will improve the user's experience.

By placing the tracking camera of the Leap Motion on the Oculus Rift the volumes of their separate tracking systems can be calibrated to match in the real and virtual world. This can be achieved by using a mount and software released during the development of this thesis.

The move of the Leap Motion from the table to the HMD was an unforeseen requirement found at the end of development. The change in orientation had a negative impact on tracking reliability since the gestures of the sphere were poorly adapted to this new position of the camera.

The developed application "Control Sphere" got a very positive response from user evaluation and has seen more than 200 downloads. The gestures combined with the visual feedback was seen as an engaging method of interaction. A version of the application was created for a regular monitor and user tests confirmed that performing the gestures in VR was a more natural experience.

Contents

Chapter 1

Introduction

This thesis provides insight into the development of Virtual Reality(VR) applications with the purpose of introducing users to a new gesture interaction platform. Gesture-based interaction and VR are two rapidly evolving technologies with a great potential to complement each other. VR benefits from gesture-based interaction since the hand tracking allows the user to see and use virtual representations of their hands. Gesture-based interaction benefits from VR by the possibility it provides to visualize the interface where the gestures are performed instead of on a screen in front of the user.

The thesis was a collaboration with Erghis Technologies AB. Erghis is a Malmö based company founded in 2014 focused on developing a gesture interaction platform. The gestures used are based on a cognitive model of holding a sphere in your hands and tapping on its surface [1].

1.1 Goal

The goal of the thesis is to use the potential benefits of combining VR and gesture detection to introduce new users to the Erghis Sphere gestures.

Chapter 2

Technical Background

The thesis aims to combine VR with gesture detection together with Erghis technology in a single application. This chapter provides insight into these technologies and the unique design aspects of such applications.

2.1 Virtual Reality

By definition, "Virtual reality is the use of computer technology to create the effect of an interactive three-dimensional world in which the objects have a sense of spatial presence [2]." To achieve this effect different technologies are used to interact with the human senses. A core part of most VR systems is the Head Mounted Display (HMD) which gives the user a view of the computer generated environment.

Figure 2.1: A person wearing the head mounted display Oculus Rift CV1 (Consumer Version 1)

By tracking the position and orientation of the HMD the display can visualize the virtual world with a point of view that matches any head movement. To send the correct image to each eye the system needs to know the exact position of the eyes in relation to the screen and optics of the HMD. The tracking can also be used for the position and orientation of headphones and make it possible to experience the sound of the virtual environment in 3D. By using the HMD as input and output, the user's awareness of head

position and sense of balance is aligned in the real and virtual world and confirmed by vision and hearing. If the alignment of the real and virtual world is done at such fidelity that our senses are unable to perceive any differences it's possible to feel present in the virtual world on a subconscious level. This sense of presence is the goal of many VR technologies [3].

However, it is very hard to achieve presence and very easy to break it. For example if the user touches something in the real world that is not present or properly aligned in the virtual world there is an obvious mismatch between our sense of touch and vision. How the brain interprets information when there is a mismatch between senses depends on how reliable the information is perceived to be. In most cases our sense of touch and proprioception is deemed more reliable than vision. So if you sit down and feel the floor with your feet about 1.2m below your eyes, your brain will not trust visual information telling it that the eyes are 1.7m above the virtual floor. In an attempt to match the differing information it might instead assume that everything at eye level is 1.2m above ground, resulting in the virtual world being interpreted as smaller than it actually is [4].

Many mismatches between senses will unfortunately give a more severe effect than the scale of the world being wrong and can easily result in nausea [5]. A common case is when it takes to long between head movement and update of the image of the display. When this happens the visual information lags behind the sense of balance and head movement causing motion sickness. The brains own feedback loop between these systems is very fast which in turn demands a stable high frame rate and low latency on the HMD. To mitigate the effect of motion sickness the vision of the virtual environment should match the user's other senses. To control where the user is looking independent of head movement is an example of severely mismatching information. Motion sickness is also a large problem when dealing with locomotion in VR. Moving in the virtual environment while the user is stationary will also cause mismatches between the visual system and the user's awareness of the body. Accelerations and changes in elevation are best avoided. When acceleration is needed instant jump to target speed is better than a slow increase or decrease, this gives the user one change to get accustomed to instead of a continuation of changes [6]. Another mitigating technique is to give the user stable points of reference that moves with them when moving through the virtual environment. Like moving a virtual vehicle with the user in it instead of just moving the user. In these cases a part of the visual information matches the other senses [7].

Another mismatch that can cause nausea or discomfort is when the users have a virtual body that behaves different than their own. This is often more severe if the user first feels like they inhabit an avatar that then goes against their expectation [6].

2.1.1 Simulating Vision

This section explains some aspects of human vision and the HMD relevant in understanding how the simulation works.

- Field Of View (FOV): For HMD's it is good to take up a large a percentage of the

user's vision to closely simulate how we view the world. The HMD's FOV depends on the area of and distance between the display, the lenses and the pupils. The computer generated image needs to match these physical measurements or the user's vision would be distorted [6].

- Perceived resolution: The resolution of a display is measured in number of pixels. The way resolution is perceived is the number of pixel per degree of vision. This means that for a given screen the resolution is perceived as lower when the FOV is larger. This leads to a trade off between FOV and resolution and that achievable perceived resolution for HMD's is much lower than for a TV, computer screen or smart phone. Techniques like anti aliasing becomes more important to blur the jagged edges of virtual objects. However more pixels means more graphical elements to process for the computer and as noted frame rate is very important [8].

- Eye Movement: The visual system can perform a continuous motion called "smooth pursuit" to keep a moving object focused on the fovea. The eye is also capable of stabilizing relative to the environment with a reflex, called Vestibulo-ocular movement [9].

- Frame Rate: The computer can generate and display a number of Frames Per Second (FPS). Due to how the eyes smooth motion is relative to both the screen and to the virtual environment the frame rate needs to be very high for any disconnects to be imperceptible [10].

- Motion to Photon Latency: Motion to photon latency is the time it takes from a motion from the user until the pixels on the display change. Even at as little as 20 milliseconds the brain registers if the proprioceptive system and visual perception are out of sync which can lead to motion sickness. Included in this time is how long it takes to get information from the sensor to the computer, to process the frame and to draw it on the screen. However a technique called Time-Warp improves this by retrieving rotational information one additional time after the frame has been processed and performs computation to redirect it before drawing. This gets the effective motion to photon latency to be shorter than the time it takes to process the virtual world to render a frame [11].

- Pixel Persistence: A frame displayed to the user is only correctly positioned the instant it is shown. When a virtual object is rendered on the screen and the screen moves while still emitting light, the virtual object will move a small fraction until the screen is updated. The virtual object was not supposed to move but it appear to as a consequence of the user moving the head. Due to the vestibulo-ocular reflex it is possible for the user to notice that the virtual object in focus does not stay sharp as the user moves the head and the screen.

  Low persistence displays have the ability to turn off the pixels between frames, leaving the screen black until the next frame is ready. This means that the virtual object

does not have time to be moved together with the user's head, and the user is shown a black screen instead of a misplaced object. When the visual system processes this information the black screens are not registered by the user. This leaves a sharp image for the user even during head movement [10].

- Deactivating Pixels: For some screens pixels are turned off to show a black color. This causes the pixels take slightly longer to change color than for activate pixels and they might not achieve the correct color during the time a low pixel persistence frame is active. Combined with the effect of the vestibulo-ocular reflex this can cause an apparent black smear around black areas on a screen when looking around with an HMD [12].

- Frame Buffer: Before the pixels are sent to the screen they need to be defined virtually in picture called frame buffer. The frame buffer is combined from pictures generated by virtual cameras for each eye. Different computations can be performed on these pictures before the final image is sent to the display.

- Stereo vision and convergence: Due to having different positions each eye has a different FOV. but there is some overlap. In this overlap the eyes can converge to see the same object from different angles. This is one way humans perceive depth. As the HMD has to create two images these can be displayed on two screens as long as the images are synced. By using two screens the screens and lenses can be adjusted together to be positioned in front of each eye. It is also possible to achieve a higher combined resolution than it would be on a single screen [6].

- Lens Distortion: The lenses are used to bend the light to set the visual position of the display at a distance from the eye that is comfortable to focus on. The lenses distorts the image causing a variety of artefact's such as chromatic aberration and a pinch cushion distortion. Instead of using a complex self correcting lens setup the computer can perform calculations on the image buffers to inverse the distortions. As the size of pixels in the middle of the image buffer are magnified by these calculations the image needs to be rendered at a higher resolution for the magnified pixels to match the size of the pixels on the display [6].

- Accommodation: Accommodation is how a single eye can focus on objects at different distances. However the lenses can only position the screen at a fixed distance and the screen does not have depth. This means that the HMD is unable to simulate this depth cue [6].

2.2 Input Methods for Head Mounted Displays

A big challenge with interaction when using a HMD is that the user can't see the real world. Meaning they can't see any controllers and they can't see their own hands.

A simple approach to this problem is to use handheld controllers, such as a gamepad, with any point of interaction within reach of the users fingers. The idea is to use methods of input that doesn't need to be seen. However if the controller is represented at the correct position in the Virtual World the sense of presence can be reinforced by proprioception of the hands holding the controller in the position it can be seen. For example a steering wheel for a driving game might be recreated in the virtual world. The challenge is then to calibrate the dimensions and position of the controller in the virtual world with regards to the position of the HMD. If the controllers are tracked in the same coordinate system as the HMD any controller can be positioned correctly in VR. By placing this type of controller in the user's hands it is possible to reach into the virtual environment and have any motion of the controller be visualized. In fact the three largest HMD's targeting consumers on release, Oculus Rift, HTC Vive, and Playstation VR, has all announced a pair of tracked controllers held in each hand to go with their HMD [13, 14, 15].

Figure 2.2: The HTC Vive, Oculust Rift DK2 and Playstation VR)

2.2.1 Gesture Recognition

Another approach to input is to track the user's body and recognize specific gestures as commands. The touch screen on a smart phone is an example of an interface that uses gesture recognition but in this thesis the focus is on touch-less in air gestures. This can be achieved in a number of different ways. For example hardware units that track rotations of limbs or camera system tracking special markers placed on the body.

The system used in development of this project is called the Leap Motion [16]. It uses a unit that includes 2 cameras that capture infrared (IR) light and a few IR light sources. By using the IR light values of individual pixels as a 3D position their software attempts to match a model of a hand to the point cloud. The cameras work at a frequency of 120 FPS and their algorithm for tracking is fast enough to keep up and provides a very low latency tracking of the hands.

The main drawbacks with the system are the limited tracking volume and the poor tracking reliably. It can only track what it can see so if parts of a hand is occluded or outside

the cameras FOV tracking either fails or has to be approximated. Many approximations are also needed to keep up with the high frequency that the images need to be analysed at. This can result in the virtual hands behaving differently than the hands that the Leap Motion attempts to track.

When using an HMD together with the Leap Motion two different tracking systems are involved. The tracking of the HMD and the hands needs to be calibrated for the hands to appear in the correct position. During development of the Erghis interface a mount was created to place the Leap motion on a Head Mounted Display. With this it is possible to know the tracking spaces position in relation to one another.

Figure 2.3: The Leap Motion mounted on an Oculus Rift DK2 (Development Kit 2)

2.3 Human Computer Interaction

Human Computer Interaction (HCI) is the design of the communication between a human and a system. The communication has two key challenges for the user, to understand how to act upon the system and understanding the state of the system and how it reacts to actions [17].

The recurring interactions are part of a feedback loop of four parts.

- The user's mental model of the system
- An action on the system taken by the user
- The rules applied on the action by the system
- Feedback to the user from the system

The loop can be seen as a conversation between the user and the system. This conversation is an important part of the user experience. When beginning the conversation

the system can gradually introduce the user to how it works and give information about possible actions to take using hints and affordances. When the user performs an action the system responds with feedback. The feedback informs the user how the action was interpreted to reinforces a desirable mental model of the system. The feedback is what allows the user to better understand the system by performing the action. The flow of this conversation is a vital part of the design and has a large impact on the user's experience and perception of the system [18].

Finding all of the loops and making sure that each part of the conversation is clear and provides a good user experience is a valuable tool when designing and examining an application [19].

2.4 Design of Virtual Experiences

A fundamental difference when designing for VR compared to other computer applications is the lack of a fixed visual screen. The main reason this is a challenge is not the techniques that are no longer applicable, these techniques can still be used on virtual screens, it is the limitless design space Virtual Reality opens up. And while anything is possible to visualize it becomes a challenge to find the possibilities that work well. But to deal with the surrounding world as a design space is a great advantage of virtual reality since human cognition is very good at spacial thinking. By arranging information in space comparisons can be done visually instead of relying on memory and internal models of relations [20].

A common use of the screen is to integrate a Heads Up Display, (HUD). It is possible to put the HUD on virtual screens, but in many instances there might be a way to naturally integrate the information into the environment. So rather than designing for a screen with buttons it is design for a room with tools [21]. Another effective concept in VR is to think of a bubble of interaction within arms reach of the user. Information and interactable objects can be positioned within this bubble as the user moves through the virtual environment the bubble follows [22, 23]. These objects will then have the additional function as points of reference for the player to mitigate motion sickness. The concept is mainly targeted to tracked input methods with 6 Degrees Of Freedom (DOF). With this type of controller the 3D area surrounding the player can be interacted with in the same way a touch display can interact with a screen. One important aspect to take into consideration is how comfortable the movements required are over an extended period of time. Any uncomfortable static positioning of the user's arms should be avoided and an extended use of large movements can be tiring and uncomfortable even though they might be natural and engaging. One approach to shrink the size of movements is to use the 6DOF controller as a laser pointer and to only require pointing at an object to select compared to touching it. The laser pointer is also very effective when interacting with virtual screens and can for example be used to navigate menus like a computer mouse on a screen [24]. Another approach is to shrink the objects you interact with. A good example of this is the Worlds in Miniature where the user in a virtual environment gets a small copy of that environment in front of them to interact with [25].

One important concept for VR is to use contextual interaction. Meaning that where in the virtual environment you perform an action affects the result. This goes well together with the concept of integrating any HUD into the virtual world. It is like using the computer mouse as a method of input, depending on what object the cursor points to the result of the left and right click is often different. Another example is computer games with a button dedicated to interacting with an object targeted by looking at it, or a button to use the object equipped by the avatar [23].

2.4.1 Design of Gesture Interfaces

Gesture recognition comes with its own set of challenges. A big part of this is conveying the set of viable gestures to the user. Since gesture recognition allows users to move their hands without restriction, it needs to be clear to the user what specific gestures can be recognized by the system. And as for any interactive system the user needs to understand what happens when an action has been performed with an obvious connection between the specific action and the result. So while the user is continuously moving their hands, the system needs to convey the whole span of a gesture, from start to finish. Meaning that it needs to be very responsive and that gestures which takes longer to perform needs continuous feedback from the system. On the other hand for small discrete gestures the user needs to understand when a gesture is about to be recognized by the system and measures should be taken so that they aren't accidentally recognized without user intent. For cases when all instances of this happening can't be avoided it should be very easy for the user to undo the action [26].

For gesture interfaces a big part of the user's understanding of the system is information about how the system understands the user, like information about how and where movements are tracked. The Leap Motion can provide this by continuously showing a 3D model of how it interprets any tracked hands.

By showing the model of the hands many simple gestures such as making the hand model touch a virtual object become easily understandable to the user, especially in VR where there is no disconnect between the position of the user's hand and the virtual representation of hand.

When designing the gestures it is important to take the limitations of the tracking in mind and making sure that the gestures are reliably recognised. For visual based tracking taking occlusion and FOV of the cameras into consideration allows the designer to understand what gestures might work poorly as the more of the hands the camera can see the easier it is to track. Examples can be seen in figure 2.4.

The concept from Erghis attempts to limit the user interactions by having all interactions focusing on the sphere shape.

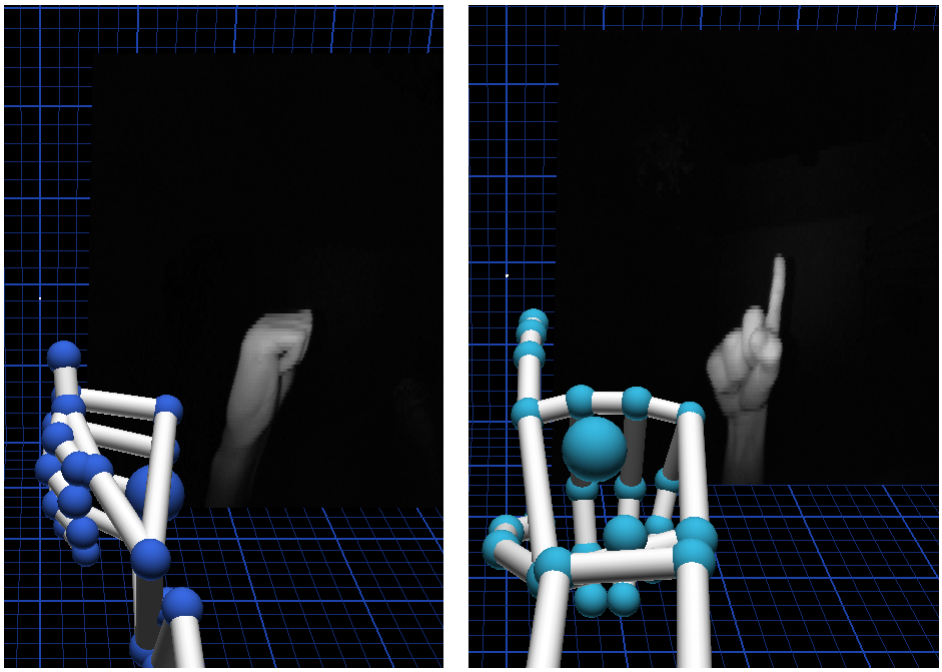


Figure 2.4: The Leap Motion tracking a user giving a thumbs up and pointing forward and the images it bases tracking on

2.5 Erghis Technology

The first program created by Erghis is called ErghisFirst. In this program the gestures were defined and used to emulate every symbol and key on a keyboard as well as a mouse. The hand positions used in this program had the palms of the hands facing the Leap Motion hand tracking device placed on the table. The detection of gestures as well as the ability to control the windows operating system with Erghisfirst was built into a Dynamic Linked Library, DLL, file.

Figure 2.5: The Leap Motion tracking device in use with a monitor

The defined gestures are as follows:

- Base position: With the Leap Motion placed facing up on a surface in front of the

user. Both hands are held out above the Leap Motion with palms facing down. Thumbs not in contact with the hand.

- Mode shift: A rotation in the wrist up or down. There are three positions defined for each hand; up, mid and down. Making a total of nine modes.
- Thumb shift: Pull the thumb in for contact with the hand. This creates a total of four possible thumb shifts for each mode.
- Tap: Any single finger besides the thumb making a swift motion a short bit towards the palm and up again.
- Fist: One hand makes a fist to access the mouse. Moving the fist in the horizontal plane from it's original position moves the mouse.

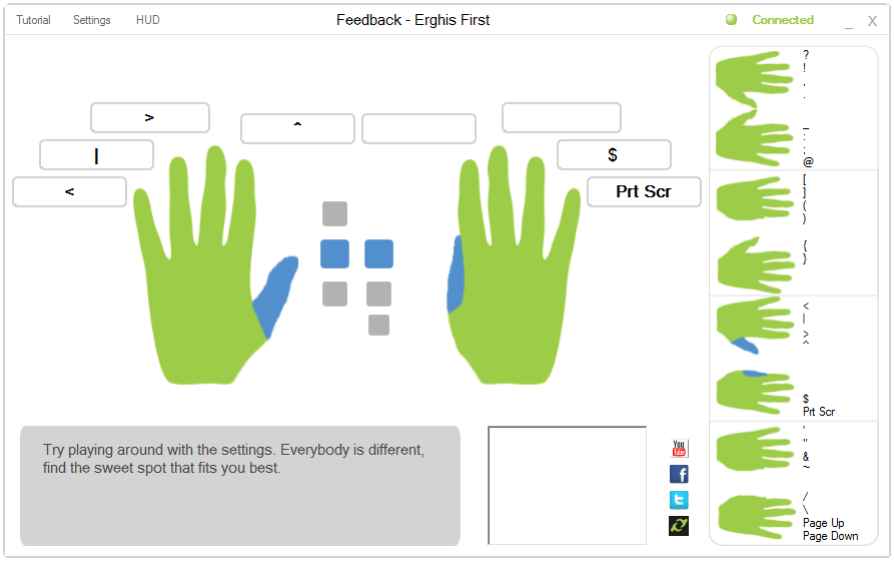


Figure 2.6: The interface of ErghisFirst

To type with the application the hands are positioned in a specific mode and thumb shift. A symbol or text is then displayed in each the boxes close to the fingers in the gui. By tapping the corresponding finger the symbol or text is sent as an output command through the windows operating system.

A second application, called Erghis 3D Sphere, was developed as part of a thesis by Chris Shields. The thesis compared a 2d and 3d interface with the same gesture detection functionality [27]. There was a change to the rotations of the palms for the Erghis 3d sphere to make both hands face each other. The application has full support for symbols and commands but no support for mouse control. The layout of 3D sphere can bee seen in figure 2.7. The user is intended to mirror the position of the hands around the sphere. The sphere was divided into two halves and when the user performs a mode shifts the hands

are animated to rotate the corresponding half of the sphere. The hands are also animated when taps or thumb-shifts are detected. The symbols of the interface is also updated with mode shifts and thumb shifts.

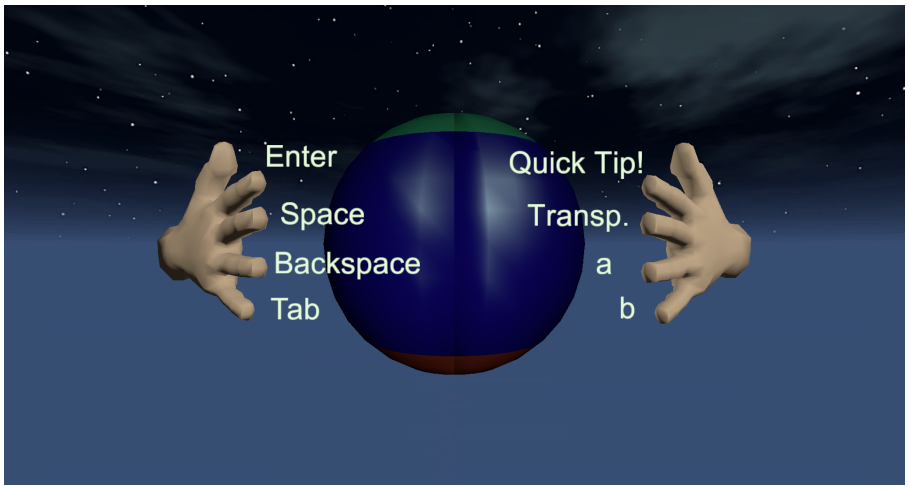


Figure 2.7: The Erghis 3D Sphere, an application for text input.

Chapter 3

Approach

To achieve the goal to introduce new users to the Erghis Sphere gestures VR applications were developed and evaluated. The applications are evaluated based on the user experience and how well it teaches the gestures to the user. In the evaluation we also investigate the benefits of adding VR to the gesture detection system by having users try both a VR and monitor based application and comparing them.

Two distinct applications have been developed for this thesis, VR Typing and Control Sphere. VR Typing was intended to be the only application but it was not providing an acceptable user experience and was instead used as a prototype which informed the development of Control Sphere. User tests were focused on Control Sphere and the result of these tests were used as a basis for an updated design.

Creating these applications were the large iterations on the design but throughout the project smaller iterations were performed in the style described in fig 3.1.

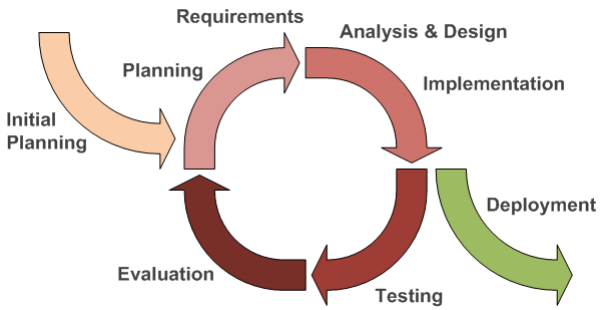


Figure 3.1: A common software development iteration cycle

The software development process can benefit from integration with user-centered design as these disciplines focus on different challenges when creating a piece of software. In user-centered design the main focus is on how the user will experience the product and in software development the focus is on implementing the software to work as intended [28].

With user-centered design unnecessary implementation time can be avoided by iterating on the design using simple prototypes and user testing early in development to find the

designs that are worth implementing. With a focus on the user experience during the software development process the developed system will be better suited to the user's needs [29, 30].

3.1 Planning and Documentation

Since the process is very flexible, a structured work flow and a way to document ideas and progress can make it much easier to maintain focus on the goals of the project. I created a design document with a simple product backlog with list of what I should work on and what I had worked on. The design document included a separate spreadsheet with all the design ideas and relevant information I read or found through testing. The product backlog was updated daily and reviewed on a weekly basis to make sure that progress was being made in relevant areas and that there was a plan for the coming week in line with the end goal of the project.

Part of the project was done with a heavy time constraint with a deadline of one month from when it began. This part of the project was done with a heavier focus on planning. More features were defined in the beginning with time and priority level dedicated to each feature. This was used to evaluate the feasibility of an intended project. If the required features of a project had a too high time estimate, the project needed to change or it was scrapped. After a project had been selected the project plan was then used to maintain the backlog.

The methodology is a simplified version the Scrum methodology used in software development [31]. The reason for a simplified version is that much of the functionality of Scrum is in place for working in teams.

3.2 Evaluation

Evaluation is performed throughout the process in a number of different ways to match the development needs.

3.2.1 Software Testing

Software testing is an integral part of development and is done to verify that the program performs as intended. A large part of development time was spent finding the source of different errors. To reduce the time spent searching in the code, the possible causes of errors should be reduced. One effective approach is to verify that every modification or addition to the program performs as expected before starting on the next. This approach makes it easier to keep track of what parts of the code is involved when an error occurs.

For this project tests were performed by running the program and performing the recognized gestures. Information was then gathered by looking at the program response and console messages. The console prints errors and warnings from the game engine and debug messages written into the program.

3.2.2 User Testing

User observation was the main method of testing used during development. This method of evaluation is performed by allowing the user to freely use the system while being observed by an investigator. The method is useful as a tool to evaluate how well a design supports user goals and tasks [28].

For this project observations were performed by allowing interested users to explore the system for as long as they liked, the visual information of the system was displayed on a screen. The Oculus Rift and Leap Motion setup was explained and the test participant was free to ask questions of the observer. The focus of the tests are the details of the participants actions. The observation is followed by a dialog with the user about the experience to better understand the user.

To get collect additional data on the value of using the application in VR compared to on a screen and to get better documentation of how easy it was to understand and use the gestures the observation was coupled with a questionnaire with more focused questions.

3.3 Tools

For this project we use the game engine Unity 4 [32]. The game engine is a software platform which makes it easier to create and manage a virtual environment with a large number of different tools and automated processes.

The HMD used is the Oculus Rift Development Kit 2 (DK2). For hand tracking the camera based system Leap Motion is used. Both of these hardware units have their own software platforms and both of these platforms have support for Unity. In both cases the support includes plugin DLL files and several examples with useful Unity assets that can be used freely in development. During my development Leap Motion released a mount to place the Leap Motion on the Oculus DK2 together with Unity assets that combined Oculus and Leap Motion functionality for use with the mount.

I had full access to the source code and Unity project from the Erghis 3D Sphere, as well as the 3D models used. This included C# code and assets to interpret input from Erghis to animate the sphere and select a text to show beside each finger inside Unity to match the output of the Erghis system.

The programming language used was C# and I used Visual Studio 2013 as my programming environment. For version control I used git and several local copies. For documentation I used Google Docs. 3D models was created in 3DS max and images were edited in Photoshop. Audio editing was done with Audacity. Videos were recorded using Fraps.

Chapter 4

The VR Typing Application

In VR Typing the gestures based on the Erghis Sphere are used to type on a virtual screen. The goal of the application was to introduce the users to an interface using these gestures combined with Virtual Reality. The final version of VR Typing is a high fidelity prototype running inside the Unity engine.

4.1 Setting up for Development

Since there was a 3D interface developed to start with I decided to use that as my basis for the VR program. I started by adapting the Unity project to use hands controlled by the Leap Motion. The Erghis 3D Sphere had previously used a static position for the hand models as Leap Motion had not provided this functionality during its development. During this development I created a project plan and started documenting my work.

Next I got the Oculus working and started to re-purpose the project for VR. I continued to stabilize the development environment. I updated to the latest available versions of Unity, Leap Motion, Oculus and Erghis and managed to find and removed software bugs from the old project that were causing crashes and a lot of latency for the Leap Motion. This resulted in a working prototype that served as a proof of concept for the project.

4.2 Early Prototypes

I began by creating a design document in the form of a spreadsheet. To create the document I researched gesture based interfaces, brainstormed with the Erghis team and tested other available software for the Leap Motion.

The design document included:

- User goals
- Questions regarding the previous applications
- Strength and weaknesses of the existing application

- Feature ideas
- Time estimates - To know when I don't think I can do the project
- Constraints - To help avoid potentially time consuming features and focus development
- User challenges
- My mental model of the application
- Design goals - To help me articulate what I wanted to achieve
- Ideas for environment design
- Snippets of relevant research

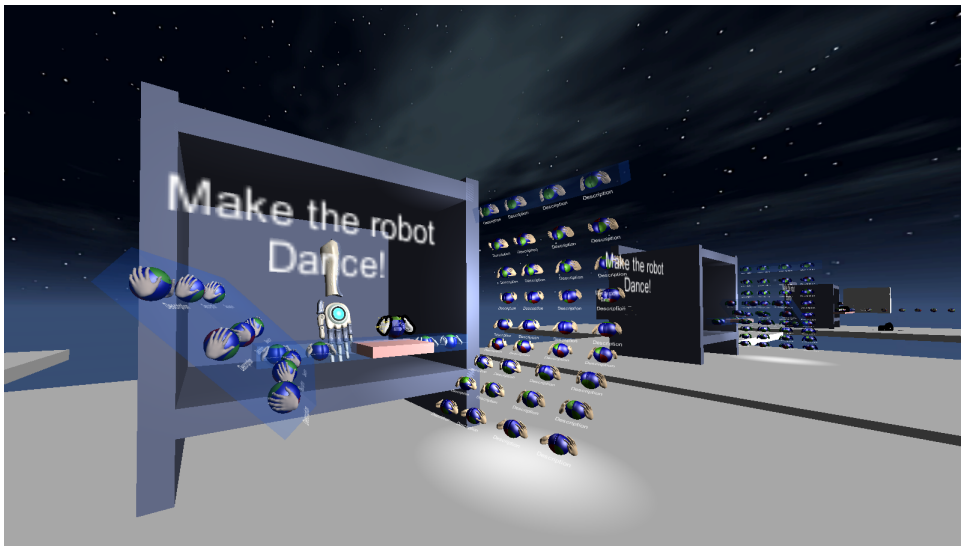


Figure 4.1: A workspace prototype with other prototypes seen in the background

For prototyping I decided on scenes created in Unity where I had a general outline of the placement of objects in the scene and some examples of different sphere designs displayed in Unity. The prototypes are quick experimentation with the layout of the workspace. The idea is to use them as a reference of the design and iterate on different ideas for the design. When evaluating the prototypes the virtual Oculus camera was positioned in front of the spheres and used to look around the environment. In figure 4.1 there is the line "Make the Robot Dance" this was an idea I had for a user goal for the application and how to convey it. As the project began it was not decided that it would be a typing application. The important aspect was that the application should teach the user how to interact with the Erghis gestures. But the general ideas explored in prototyping the environment was workspace with a point of focus in front of the user. This mimics the setup of a desktop

with a screen in front of the user and the Leap Motion. When I realized this was the case I decided to embrace it and created a virtual screen. I intended for the user to type messages which makes the most use of already developed functionality for typing. Fourteen different prototypes were developed for the scene, examples can be seen in figure 4.1 and 4.2.

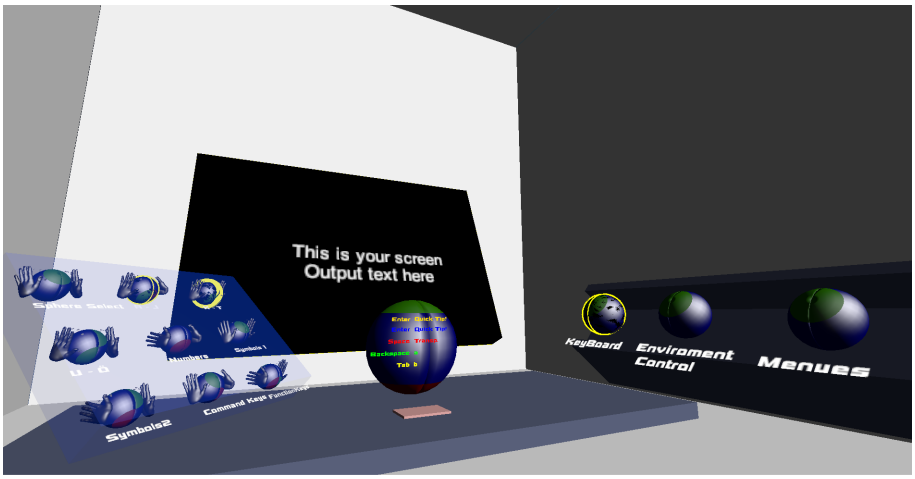


Figure 4.2: A prototype layout for VR typing

When prototyping the design of the sphere I focused on conveying the points of interaction and different experiments with the placement of information. I kept the two half spheres and the base color scheme as this had been adapted by Erghis as the way to convey modeshifts to the user.

In the Erghis 3D Sphere the thumb shifts were visualised by an animated thumb, but the animation of the hand model had been changed to be handled by the Leap Motion software. I attempted to show the thumb shifts with a 3D version of the ErghisFirst design, where the color of the thumb is different when a thumb shift is registered. It did not fit the rest of the hand and I found the interaction problematic as moving the thumb doesn't match the concept of tapping on a sphere very well and adds a lot of complexity for the user. The benefit of the thumbshift is that it multiplies the available options the interface provides by four, but it was not certain the additional options would be needed. There are 90 options remaining after the thumb shifts are removed and that is enough to create a new set of commands for typing and many other applications. After discussing this with the Erghis team the thumb shift was removed all together.

4.2.1 Collaborative Evaluation and Brainstorming

In a meeting with the Erghis team we evaluated the potential merits of my suggested features. We realized that it was not obvious if all the needs of the interface was covered. To make this easier we decided to find all of the interaction loops of the design with the user action and system feedback. We listed all the actions the user can perform and tied them to the features of the interface that communicates the action and its feedback to

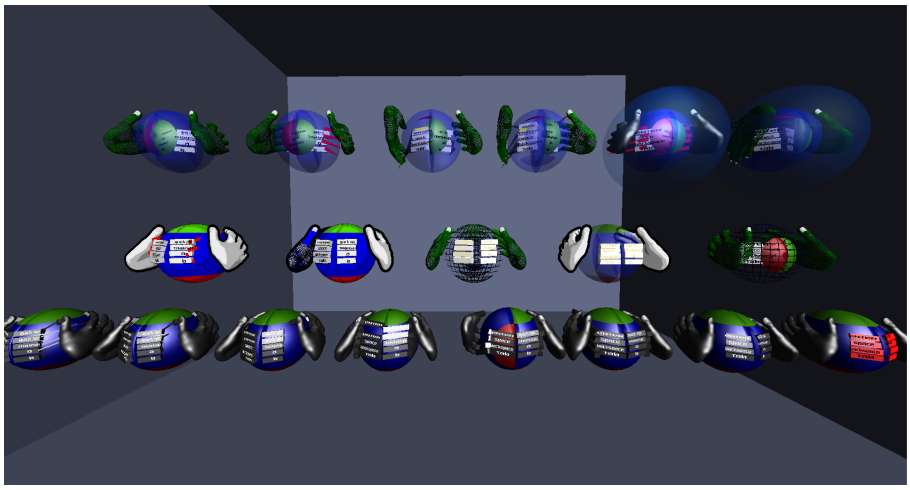


Figure 4.3: Different sphere design prototypes

the user. In figure 4.4 the first row lists the different actions alternated with the feedback

Grab sphere	Feedback	Twist	Feedback	Tap	Tap association	Feedback	Navigation of menus	Feedback
Ghost hands	Handsappear	Separated halfs	Updated text	Ghost animation	Color coding	Colorchange	A mission for the user	Lightup of menus
In focus	Sphere moves	Table of rotations	Color change	Color coding	Placing text inside	Sound	Removing thumb menus	
	Ghost hand disapear	Ghost hands	Lightup menues			Output	A table of options	
	Sphere scales		Sound					
					Scrapped	Scrapped		
					Lazers	Wave		
					Mechanical connection	Particle effect		
					shadow from finger			

Figure 4.4: The first row lists the different actions alternated with the feedback the system should provide the user. The other rows has the features corresponding to the actions are listed

the system should provide the user. The other rows has the features corresponding to the actions are listed. The features are described in more detail in chapter 4.4.

4.3 Implementation

Following the brainstorming the feature list was set and I started to work on developing the software. Besides the features there were more general requirements. It should work as intend with the latest available version of the Erghis platform. It needed a high performance to be suitable for VR. The reaction from the sphere had to feel natural with regards to its position in their hands.

4.3.1 Positioning the Sphere

The hardest technical problem to solve was the continuous placement of the sphere.

At the start of development a simple solution was implemented with the sphere placed at the center point between the hands and many aspects of the design was developed with this solution in place. This solution only looks good when the hands face each other. For example if both hands are facing up the sphere will still be placed between them instead of above them.

Too improve on this solution I worked with sketches of possible hand positions and orientations and tried to find ways to position the sphere according to one set of rules and requirements. When a set seemed to be a possible solution I would implement it and view the results. The first requirement was that the position should depend on not only the position, but also the orientation of both hands. The movement of the sphere should also be fluid without jumping between positions.

In my first attempt used a ray along the normal of each palm and placed the sphere on the point between these rays where they were the closest to each other. This doesn't work well when the rays are close at points close to one hand but not the other. Some attempts were made at adjusting the sphere position in a different way when the point is to close to one hand but the distinct change breaks the fluid motion of the sphere as it then jumps from one position to the next between frames.

The solution I found was to add another requirement were the sphere is always at an equal distance from both hands. To implement this I made use of the fact that all points at an equal distance from two fixed points are positioned on a plane. The plane is orthogonal to the vector between the fixed points and can be constructed by using this vector as a normal and the center point on the vector. I can then intersect rays from the normal of each hand with the plane and take the point in the middle of the intersections.

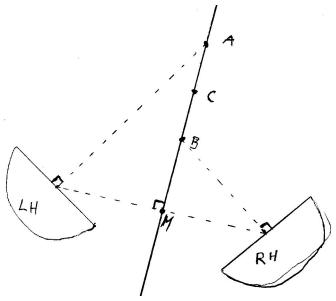


Figure 4.5: A 2D version of the points and lines used to find the sphere center. C is the point being calculated and LH and RH are given. LH represents the left hand and RH the right hand of the user. M is the middle point between RH and LH. The line is placed on M with a normal in the direction from RH to LH. The line is used to get the intersection points A and B with ray's from the normal of LH and RH respectively. C is the middle point between A and B.

The calculations described in figure 4.5 are executed every frame in 3D. In 3D the line is represented by a plane and the points and vectors are in 3D. As we get a new orientation and position for the hands every frame the Line,(or plane,) M, A, B and C are continuously

recalculated. Note that every point on the line, (or plane,) is at an equal distance from both RH and LH. This distance is then used to determine the scale of the sphere.

It is possible to face the palms away from the calculated plane so the ray from the palm never intersect. When this is the case we can take a point at a distance from the palm normal and project it on the plane. This point can be used as a substitute for the intersection point while keeping the rest of the calculations the same.

This now works fairly well, however any slight shift in rotation of the hand can greatly affect the position of the sphere, making it shake even when the user is trying to keep it still. To increase stability I moved the center of the sphere to a point between M and C. The distance from M to the sphere center is capped with a max value and scaled depending on the angle between the normals of the palms. With this implementation the user can easily move their hands around the virtual sphere without any unexpected behaviour from the sphere.

4.3.2 Features

Implementing the new features went smoothly. Smaller features like color coding the design and changing the color for a short duration on a tap were quick to implement and contributed a lot.

The more time consuming features to develop was the "light-up menu", the "Ghost Hands" and the actual text editing.

The light-up menu is intended to give the user an overview of each possible option with every possible modeshift displayed with a description and the current mode shift indicated. I created a simple list of objects and mapped each modeshift to an index on the list. The menu displays nine spheres with the possible combinations of different rotation of the sphere. Below each sphere there is a description of the options shown by selecting that sphere. As a mode shift happens the currently selected sphere gets a yellow glow surrounding it. The glow around previously selected sphere fades out. I created a material for the surface of these objects and when a modeshift triggers I take the indicated object and modify a value in the material to give the object a glow. This is change is interpolated from start to end value over a short period of time and the same process is applied in reverse to remove the glow when the modeshift is changed again.

The Ghost Hands are intended to show the user the possible gestures and be a tutorial the user can follow to learn the interface. To achieve this the ghost hands needs to be animated. I was unfortunately lacking in knowledge about animation and couldn't find a way to record animations with the Leap Motion to play back later. To avoid spending too much time on learning how to animate I created simple rotations and translations in code. I created a small state machine to which transition between four different animation when the user performed the gestures shown by the animations.

After implementing the different features that communicate actions and feedback to the user, I focused on the text editor and having the text the user is typing displayed on a screen in VR. The Erghis 3D Sphere application used a system that displayed the text and symbols the user could enter from the current mode. Since the removal of the thumb shifts

the displayed symbols have not matched the output of the Erghis system as these were managed in two separate lists. I decided to parse these texts rather than using the output of the Erghis systems simulated key-press. For the input to be correct I had to edit the list of symbols to work as a text editor despite the reduction in possible inputs. I implemented the thumbs to be used as a tap to get all of the inputs I required. For the parser I checked for the special cases such as "Space", "Shift" or arrow keys and implemented these commands into my simple editor. A blinking cursor was also implemented to show where in the text the user is editing, similar to most text editors.

4.3.3 Fixes

The tracking of the system was not completely reliable and I implemented several small fixes to mitigate this problem.

The Leap Motion has a system assigning whether a hand is a left or right hand. Unfortunately this system is often mistaken which leads to the user's hand facing the opposite direction and using the model of a left hand instead of a right hand. Since VR Typing uses different color coding on the right and left hand I enforced a system where if two hands are detected, the hand to the right is a right hand and the hand to the left is a left hand, otherwise the default system is used.

I also tweaked the mode-shifts by introducing a buffer zone where the mode-shift occurs at a higher angle when the user rotates the hand up and a lower angle when rotating down.

Another fix was to step through the fingers being tapped from the pinky to the index finger and using the first tapped finger instead of vice verse. This is more often the correct interpretation since the Leap Motion is placed on the table and might have trouble seeing the fingers above and just assume its moving with the finger below.

4.3.4 Camera Mount Experiments

During development we had received a mount to place the Leap Motion on the Oculus Rift with an updated version of the Leap Motion sdk enabling tracking from this position and featured the ability to render the images from the camera in Unity.

We decided to update the sdk and test the features. Everything in the design that was not not part of the interface was removed from the scene and the new features were setup to work with the Oculus. This allowed for an augmented reality experience where the user sees the sphere in the room they are present in instead of the created virtual environment. However placing the Leap Motion on the HMD caused the interpretation of the mode-shifts and taps to be incorrect and I decided to end the development of VR typing instead of solving this.

At the end of this development the program was running inside the Unity editor with each of the features from the collaborative brainstorming implemented without any game breaking bugs.

4.4 Description of Application

This chapter describes the VR Typing application in detail, reflections on the design can be read in the discussion in chapter 8.

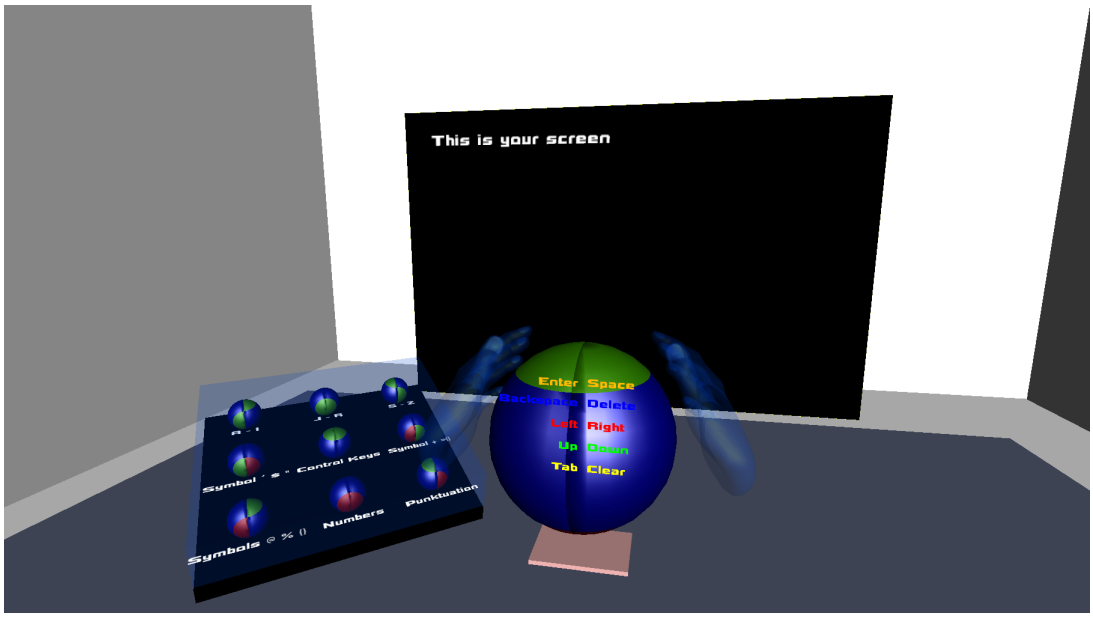


Figure 4.6: The layout for VR Typing

With the application launched and the user seated in front of a table with the Leap Motion the Oculus Rift is equipped to begin using the application. They are placed in the environment seen in figure 4.6.

The sphere is shown in front of the user with the transparent "ghost hand", (seen in figure 4.6), showing the correct hand position to grab the sphere. On each side of the sphere there is a column with five rows of text. The column on the left is tied to the left hand. The top row is tied to the thumbs and the bottom row to the pinkies. The color of the text is matched by colored objects on the fingers.

The ghost hands are intended as a tutorial the user can follow. As the user puts their hands in the Leap Motions field of view virtual hands are shown and the sphere is placed relative to both hands. As the hands moves the position of the sphere changes and the scale of the sphere updates depending on the distance from either hand to the center of the sphere. If the user removes their hands from the tracking volume, the sphere and ghost hands disappear. When the sphere is positioned in the user's hand, the ghost hands follow and starts to animate with the right hand rotating upward. If the user copies the rotation of the hand, the half of the sphere closest to the hand rotates 90 degrees in the same direction, which is notable by the color of the top of both sphere halves being different. The rotation of the sphere is accompanied by a sound cue. As the rotation happens the text displayed within the sphere is updated. There is also a change in the "Light Up Menu" seen to the left in figure 4.6.

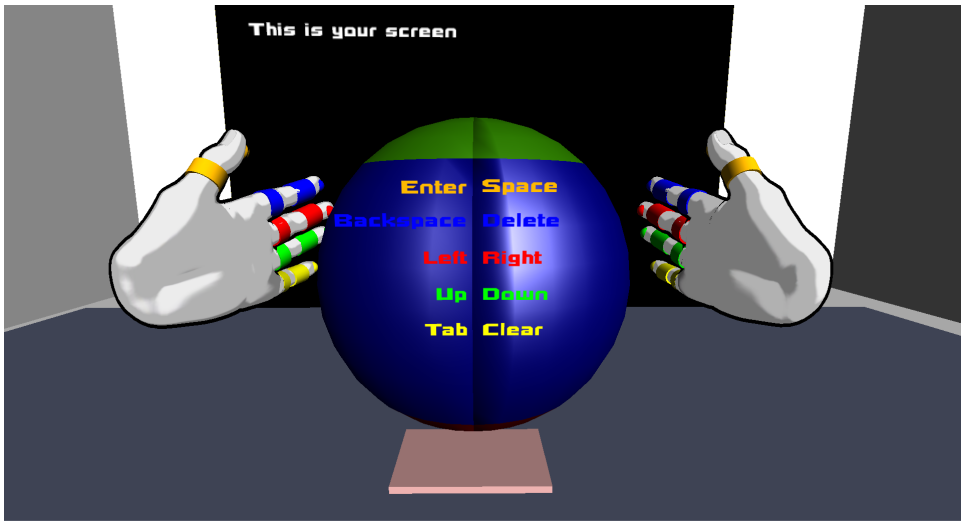


Figure 4.7: The Sphere and Hand Design

The menu displays nine spheres with the possible combinations of different rotation of the sphere. Below each sphere there is a description of the options shown by selecting that sphere. As a mode shift happens the currently selected sphere gets a yellow glow surrounding it. The glow around previously selected sphere fades out.

If the user's selected mode matches the "ghost hands", the ghost hands starts to do a tapping animation. If the user copies the tap animation the colored text connected to that finger flashes in a teal color for about 0.1 seconds. As the tap is performed there is a clicking sound and the letter appears on the screen.

The ghost hands perform one more rotation and followed by tap that the user can match. If the user match the sequence of gestures, the ghost hands disappear.

In figure 4.7 the commands Left, Right, Up, Down moves the cursor on the virtual screen. Enter moves the cursor together with any written text to the right of it or below it down one line. Clear removes any written text and places the cursor at the beginning.

The application never worked outside of the Unity editor. And since I no longer have access to Unity 4 professional, which is needed for the application to compile, there is unfortunately no longer a working version of the application.

Chapter 5

The Control Sphere Application

Control Sphere is an application using the Erghis Sphere to combine particles of different shapes and colors into a particle effect. The goal of the application was the same as for VR Typing: To introduce users to an interface using the Erghis Sphere gestures combined with Virtual Reality.

Control Sphere is an improvement on VR Typing but is a separate application due to fundamental changes in the design. The reason for these changes was the unsatisfying user experience of typing with unreliable tracking. This is discussed in detail in chapter 8.1. To decrease the impact of failing tracking Control Sphere was designed with output that had a smaller effect on the goal of the user than an intended sequence of letters.

The Control Sphere application was submitted to the "Leap Motion 3D Jam"[33]. Two separate versions was developed for a regular monitor and for the Oculus Rift. The Oculus version of the application has the option to place the Leap Motion on the table or mounted on the Oculus Rift. The Oculus and monitor versions were tested and improved upon in updated versions.

5.1 Design and Development

5.1.1 Planning the Concept

With the design for typing done I had a great platform to start out from. First thing was to figure out what to do with it. I began by starting a new section of the design document specific for the new application.

The new part of the document included:

- Design goals - To help me articulate what I wanted to achieve
- User goals broad - To have a baseline of the core intent for the user
- User goals specific - To focus each feature to a simple goal for the user
- Feature ideas

- Time estimates - To know when I don't think I can do the project
- Constraints - To help avoid potentially time consuming features and focus development
- Development challenges - To highlight potential time sinks and design around them when possible.

With the goals specified feature ideas were added until a concept was complete. The foundation of the concept was a game with a large number of magic effects where the sphere is used make a selection. The selected effect is then fired at rounds of enemies approaching the user.

After I hand constructed a complete concept I pitched it to the Erghis team. After discussion it was concluded that it was not good enough. Players were lacking a reason to select different magic effects. To have each magic affect the enemies in different ways would give the player a reason but it would be time consuming and limit the number of selections possible to make with the sphere.

After the pitch failed I decided to take a new approach and begin with prototyping interesting parts of the original idea based on the VR Typing application. I made a prototype that allowed the user to select one particle effect from a list of 50 and activate it with a gesture. I downloaded the effects, mapped them to a tap and made the text on the sphere list the names of the particle effects. When an effect was selected I showed it as active by changing the color of the corresponding text. The activation gesture I used was a check if the normals of a hand did not intersect the plane where the sphere is positioned. When activating the gesture a ball was thrown from the palm of the hand facing away from the sphere. When the ball hit the ground it triggered the particle effect. The prototype was then moved to an environment created using a terrain modifier and a skybox instead of the boxlike workspace of VR Typing.

While testing the prototype I found that the interaction had become much slower. I examined the interaction loops to understand what changes had occurred. The new activation gesture was a big gesture and the response from the system was slower with the thrown ball having to hit the environment and the particle effect taking some time to execute. In contrast the smaller loop with the tapping gesture and the ability to input a stream of commands is a very strong point of the interface. In VR Typing the problem was that this stream of commands was broken by errors in tracking that the user had to spend time correcting. To maintain the flow of the interaction the need to correct errors can be removed. With this realization the prototype was adapted into a new concept where the user can play with the interface by adding a number of different particles in varying colors to a firework effect. Unintended input will affect the resulting combination of particles but not the flow of the interaction.

The new concept did not line up with the expectation of the Erghis team. Their idea was to rework the initial concept with a focus on various effects of a limited number of spells fired at enemies. I preferred my own concept and the team agreed that the issues from the first concept was not present in the new one.

5.1.2 New Design

I decided on making a new design of the sphere for this new application. I had worked with the two half spheres rotating for about two months and had come up with a different idea I thought might work better. I began with some sketches of the new design I wanted to do and then checking of some quite simple features that would not take much time. The

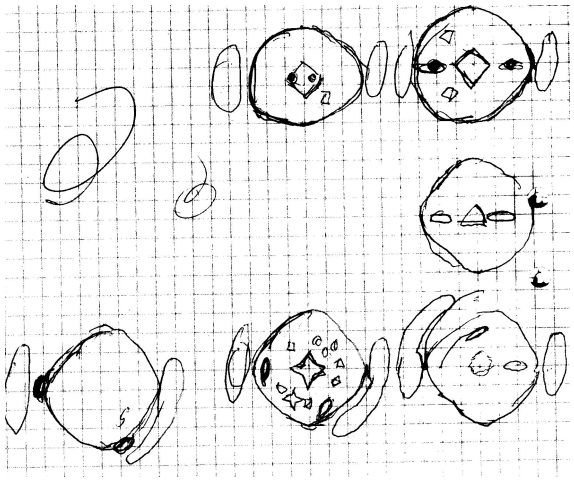


Figure 5.1: Sketches used when working out the new design

core idea of the new design was to move two objects along the surface of the sphere instead of using two rotating halves, this along with a transparent sphere allows for placement of objects in the center of the sphere.

5.1.3 Additional Gestures

When implementing the new design I reused the old animations that rotated the half spheres. For the objects moving around I used small glowing spheres. When placing the glowing spheres I got an idea for a new way of getting them to shift position.

In figure 5.2 the blue Z axis is facing forward and red X axis facing to the right with the glowing spheres placed in the X-Z plane at an angle of +-45 degrees from the Z axis. When finding these positions I made a version placing them at +- 90 degrees from the Z axis. So when the user's hands rotated around the X axis the glowing spheres moved around the Z axis. This created a cognitive dissonance and felt very unnatural. I realized that it would be easy for the user to move their hands around the Z axis to mimic the glowing spheres. I decided to implement this as an extension of the existing mode shift from Erghis. The algorithm works by placing the sphere using hand position and rotation and then using the position of the hands in relation to the sphere as a method of input. The mode shift triggers at a higher position when moving the hand up compared to a lower position when moving down. If there was a single position for both up and down and the hand would be positioned at that point, the modes could trigger multiple times. The different positions creates a larger volume for the hands to move around in without changing mode.

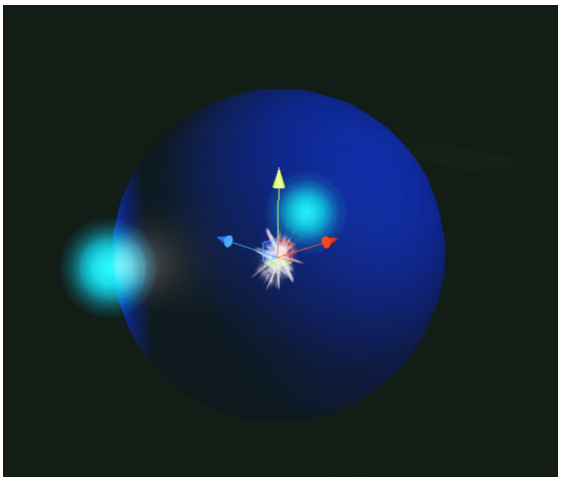


Figure 5.2: Prototype of the new design. The glowing spheres can move along the surface to the top or bottom of the blue sphere. The blue axis is the forward direction

The new combination integrated naturally with the previous gestures by having the older rotation based modeshift override the new position based.

A new gesture was needed to activate the firework. Testing had shown that the gesture used in the early prototype was difficult to understand as just inside the office people attempted to activate it in different ways. As throwing the sphere was to open to interpretation I changed the metaphor into squeezing the sphere or pressing the hands together. The gesture activates on the condition that the distance from either hand to the sphere center are lower than a set threshold. When testing the new gesture within the office it was performed consistently on the first attempt and I got very positive feedback on the feeling of the gesture.

5.1.4 Implementation

With the base design of the platform done the core of each part of the design working in VR I had a sudden change of location from the office to my home town Helsingborgs cultural center, Dunkers. This provided me with many new testers who had never tried the system. This enabled me to quickly get feedback on newly implemented features. As an example I made it possible to add several particle effects to the firework and showed this by displaying every effect inside the sphere. Feedback helped me realise how cluttered this looked. To improve it I replaced the effects inside the sphere with a single particle placed around the center on the horizontal plane as a symbol for each effect. This was much clearer but tester felt it was ”lifeless” compared to the first example and so I added an animation which made the particles rotate around the center of the sphere. I realised that the distance to the particles from the center can match the threshold where the activation gesture triggers. So when the radius of the sphere matches the radius of the particles place around the center the particles are fired.

One idea from the brainstorming session for VR Typing was a line from the fingertips to the sphere. The feature was discarded since it did not fit the rest of the design for VR Typing but I decided to try out an animated variant of it in this new design. I did this with a small beam effect from a fingertip to the center of the sphere when that finger is tapping. The effect is short, lasting for just 0.1 seconds. My intent was for it to be a subtle connection between the tapping action and the result. The feature was rarely noticed in testing but when I remarked on it the response was positive so I decided to keep it. A similar animation was added for modeshifts with the markers moving around the sphere spawns a smaller copy that quickly scales down while moving into the center particle which then changes. The results in testing were similar to the beam from the fingertips, it did not grab the users' attention but it got a positive response when I mentioned it.

I later went back to the office since I needed updates to the Erghis platform to be able to make a build of the Unity project and to get the Leap Motion to work in the mount on the Oculus. I worked out what was needed to get the headmount version to work and what caused the problems with Unity. Holger Andersson, who is the main developer of the Erghis platform, then helped me by using this information to apply the fixes to the Erghis software. With the Erghis side working I could adapt the Unity project and get the headmount to work and the program to compile. I also added the possibility to switch back and forth between placing it on the table or in the headmount.

I continued to add features to the design. It was remarked that the placement of the particles rotating around the sphere seemed a bit random and so I decided to visualize them by connecting lines between every point to form a pentagram. When users' add a particle they can now see the points of the pentagram as slots that they can fill out. When each point has a particle I wanted it to be obvious that it wasn't possible to add more, so I removed the modeshift markers and made a unique center particle effect when this occurred.

To make each registered action even more distinct I added new sound effects. I also made sure that the firework effect had a nice sound and that each explosion on the sky was synced to a sound effect.

To improve the look and user experience of the design I put effort into how the design was lit and colored. Since different colored particles are added to the points of the pentagram I made the color of the points transition to match that particle. This meant that the color of one end of two lines changes while they keep their color on the other end. Each added color was mixed into a blend which made up the color of the sphere. To make it affect the surroundings better the sphere was given a light source with the color of the light matching the sphere, this is very visible on the hands. A great improvement was giving each explosion of the firework a colored light which faded together with the particle effect. The part where it fades out instead of suddenly disappearing is important. All of the dynamic changes to the color were initially very abrupt and very noticeable, happening from one frame to the next. To fix this I modified them to linearly interpolate over time to their new color.

Environment Design

The virtual environment of VR Typing was not very refined. As the environment can have a big impact on the user experience I decided to spend more time and effort on constructing the scene for the Control Sphere application.

To begin with I did my testing in a quickly sculpted terrain. There was a platform to stand on with an open area in front surrounded with mountains and a standard sky-box. I also added water to this test scene which the testers agreed looked very nice with how the firework was reflected on the surface. I decided on using the water and thought the scene as putting on a fireworks display for a crowd in a city. I pitched this concept to my friend Viktor Högqvist-Nilsson whom helped me with creating 3d objects. To show what I meant I used images of the canal in Malmö on spots that matched what I had in mind. I decided to place the character with the back against the column of a bridge in the middle of the river for it to be natural to shoot the fireworks along the direction of the canal. The edges of the canals were raised so that the player couldn't see everything on ground level. This allowed us to work less on details. The model of the pier with the stairs took many iterations to get the size of the steps to feel right and match the height of the bridge. The skybox was replaced with a night sky and the color of the directional light was changed to a blue tint to match.

With the required models added to the scene with buildings and trees I tried it out on the Oculus. The buildings didn't feel as nice as I had hoped and had an issue with black areas on the windows. These would cause an apparent smear around the window when you looked around with the Oculus. The problem is that when the screen of the Oculus has to change a pixel from black to a lighter color it takes more than one frame. I decided that it was not worth trying to fix this issue and instead removed the buildings from the scene.

Since the buildings were removed the concept of the design had to be changed. I decided to attempt create a more mystical feeling by reusing the trees and bridges over the river in a forest area. The crown canope of the trees was repurposed as bushes by scaling and twisting them and to add more life to the scene I and created a particle effect of leaves falling.

To reinforce the feeling of the scene a matching soft background music was added.

Introduction Video and Start Menu

For the competition I had to hand in a video. I realized that this video showed all of the gestures used and this would be useful for any user to see. So I decided to record a video of using the sphere and mention the different gestures used. For the video I cleared out the scene, made the background black and added a normal game camera recording only the hands. I edited the video with captions explaining what was going on.

I had a few settings for Control Sphere that I wanted users to be able to change. The placement of the Leap Motion from the table to the mount. The direction of the Oculus camera needed a reset function. To turn music of or on as well as some graphical settings

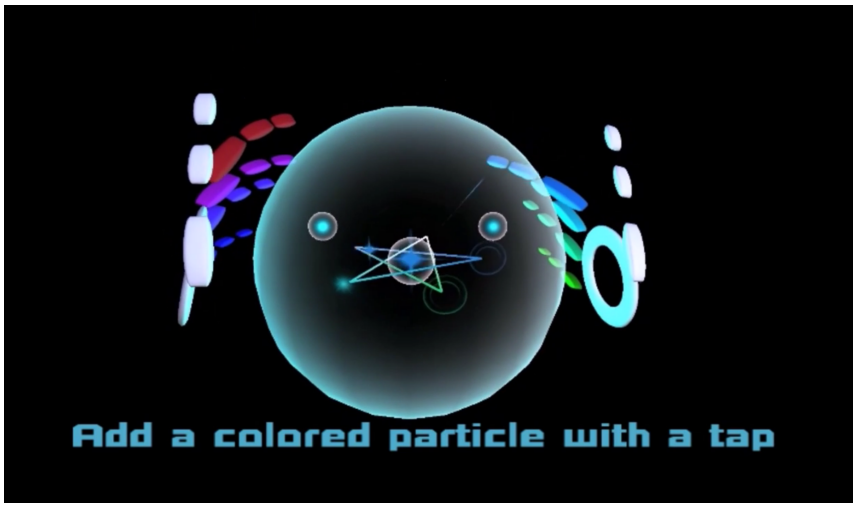


Figure 5.3: A screen shot of the Control Sphere from the introduction video

that could be changed to get lower quality visuals but better performance to allow a wider range of computers to hit the 75 fps of the Oculus DK2. I would have liked to put the settings outside of the virtual experience when starting the program but I did not have a lot of time left and so I used the first solution that came to mind and use the keyboard to toggle the settings with different keys.

To begin with I made a screen with all options and put it next to the tutorial video. After a quick test within the office we realised that a user starting up this experience would be overloaded with information. I decided to gate of the information in a few steps. I made a small room where you first set up the placement of the Leap Motion. When the selection is performed the user is brought to a screen with setting with a glimpse of the world outside. After the settings an introduction video is shown before the user is brought into the experience.

5.1.5 Monitor Version

When the application was submitted to the Leap Motion 3D jam I got a request for a version of the application for users that did not have the Oculus. This was implemented easily by replacing the Oculus controller with a camera in Unity, removing unnecessary menus and changing the height of the fireworks effect.

5.2 Description of Application

This chapter describes the Contol Sphere application in detail, reflections on the design can be read in the interpretation of results in chapter 6.2.2 and in the discussion in chapter 8.

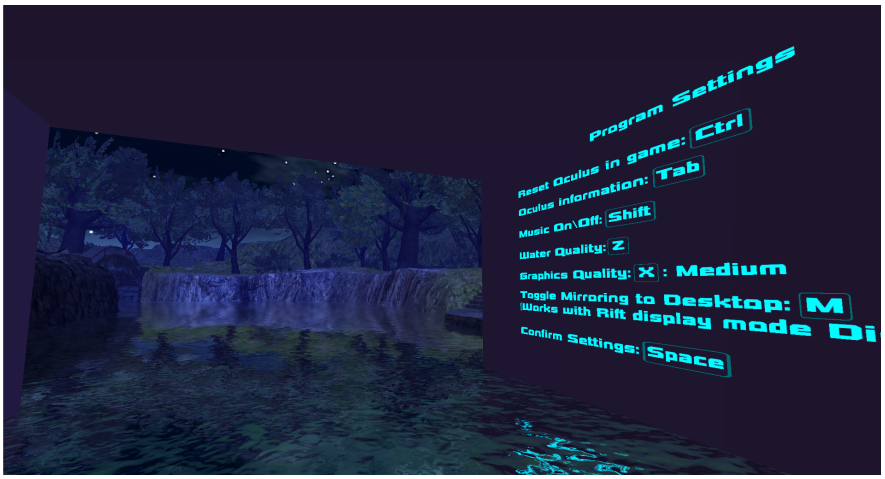


Figure 5.4: The second settings page

Two versions of Control Sphere was developed for the Leap Motion competition, one with VR and one without. These versions were used in the tests described in chapter 6. For the screen based version the difference is that the user's point of view is fixed and the start menu has no options for the Oculus Rift. The applications has sen more than 200 downloads.

A video of the Oculus Rift version of Control Sphere can be seen online at `https://www.youtube.com/watch?v=HGrG4UrabJ8`.

With the application launched and the user seated in front of a table with the Leap Motion, the Oculus Rift is equipped to begin using the application. The user is presented with a small room with the camera floating above a stone platform with water surrounding it. As the application starts music slowly starts playing. On the walls there is with a prompt to select placement of the Leap Motion with the alt key and recenter the Oculus with the space, the default placement of the Leap Motion is listed as on the table. When pressing space the user's current orientation is set to correspond to forward in the virtual world. As this is done the scene is updated with a small sound effect. The wall in front of the user is removed revealing that the water is part of a river with some hills trees and a night sky in the background. To the right the user is presented with program settings. By hitting space to continue the settings and the roof of the room disappears behind the player and a screen with a video is brought up in front. Below the video is text informing the player to hit space to continue and that space brings the menu back up and that the escape key quits to desktop. The video shows the sphere being used to select 5 particles with short descriptions of the gesture to select a shape, add a colored particle with a tap, and activate the firework by pushing the sphere into the pentagram.

If the user brings up their hands in the FOV of the Leap Motion the virtual representation of their hands are show and the sphere is positioned in them. The position and scale of the sphere is continuously following the position and orientation of the hands. The sphere is a transparent white, with a white pentagram spinning around a smaller sphere inside

it. The size of the pentagram does not change with the size of the sphere. The sphere gives of a light that is reflected on the hands and the scene. The modeshift signifiers and the selected particle are not present and modeshifts or taps can not be performed. If the activation gesture of pressing the surface of the sphere into the points of the pentagram the walls of the room and the tutorial video disappears by moving down and shrinking. The gesture is accompanied by a sound effect and an animation where the size of the pentagram shrinks a bit before it's size is restored. It would have disappeared the same way by hitting space.

The user can now fully interact with the interface and look around the virtual environment seen in figure 5.5. As the user rotates one hand up the mode shift signifier on that

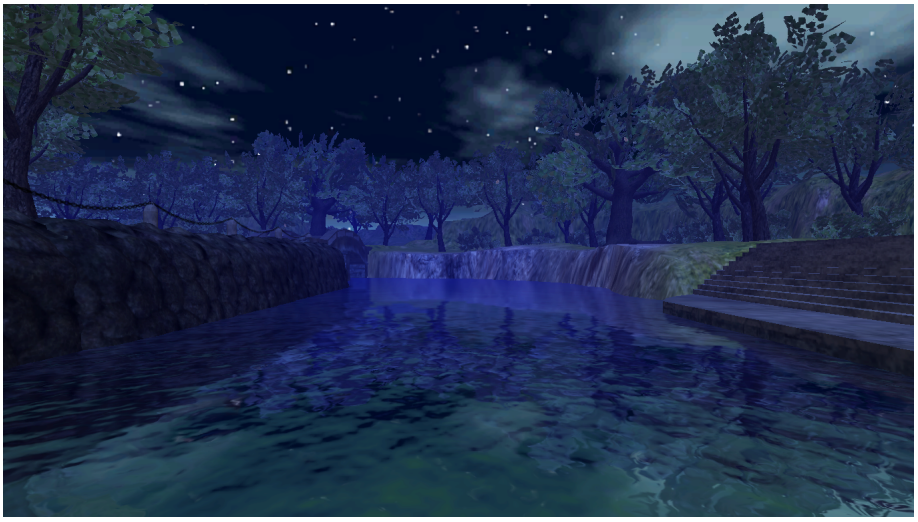


Figure 5.5: The Control Sphere environment

side of the sphere moves to the top of the sphere as it reaches the top the center particle fades and a small copy of the both modeshift signifier quickly moves to the center while shrinking. When it reaches the center a new particle grows in the place of the previous one that faded. This process takes 0.3 seconds and is accompanied by a sound. While it happens no taps can be registered occur. If the user perform another modeshift during those 0.3 seconds a new animation is started when the previous one finished.

When a user performs a tap a sound effect is heard as a small beam in the color of the tapped finger extends into the center of the sphere. As the particle in the center is hit by the beam the particle and the whole sphere changes into that color. When the color has changed a smaller copy of the particle moves to one of the points of the pentagram. As this copy moves the center particle changes back to white and the tip of the pentagram changes to the color of that particle. If subsequent taps are done with a different color the center particle changes to that color and back while the sphere is colored in a mix of all previously added colors. Each new added particles move to the position of the next point of the pentagram in a clockwise order.

When all five points of the pentagram has changed color the modeshift signifiers are

removed and the center sphere and focus particle is exchanged for a fiery effect in the same color as the sphere. As this point no taps or modeshifts are registered. When the user

Figure 5.6: The firework trail effect

performs the activation gesture a sound effect and small scaling animation is performed. It triggers a flash effect in the color of the sphere, while the flash effects fades a sound effect is played. From the flash a trail shoots out over the reflective water before moving upwards. All particles on the sphere disappeared with the flash and while the trail effect is gaining height the color of the sphere and the pentagram changes back to white. To follow the trail the user has to look up.

The moment the trail stops the firework effect starts to explode. They are bursts of many copies of the particles the user added that spread out to cover the sky. As the user added five particles there are five explosions originating from the place the trail stops distributed randomly over a short period of time. Each explosion comes with a popping sound effect and a bright light in the color of the particles that lights up the scene before fading in intensity over time. As the firework fades the mode shift signifier and center particle reappears and the user can create new combinations of firework.

Should the user perform the activation gesture with zero particles added the scaling animation of the pentagram is performed without sound effects. Is it performed with one to four particles added the firework is sent of with the corresponding number of explosions.

The user can at any point hit space, which would bring up the menu followed by the tutorial.

Should the user hit escape a credits video is brought up and played until the end or until the user hits escape again at which point the program ends.

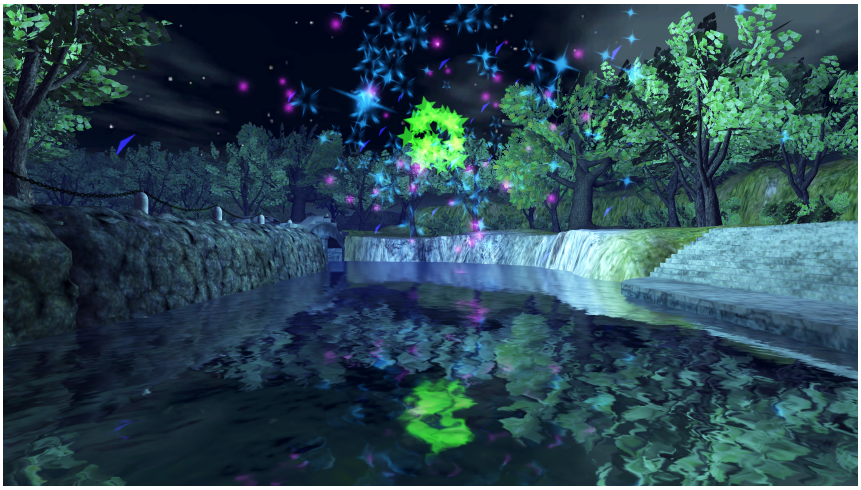


Figure 5.7: The firework effect on the monitor version

Chapter 6

User Testing

A combination of informal user observation and a more focused test with a questionnaire was used to evaluate the user experience and how well Control Sphere achieves the goal of using VR to introduce the gestures to the user.

6.1 Informal Observations

I have observed 40 people experience the version of Control Sphere submitted to the Leap Motion 3D jam. These were informal demos at VR related meetings and with friends and family. During these informal observation tests the goal was for to gauge the general impression of the experience and finding if there was any parts that were hard to understand.

The users were free to ask questions, look around and experiment for as long as they liked. When they were done I discussed the experience with the user.

These informal observation tests got a very positive response to the experience of the application. There were compliments for the look of the sphere, the sounds, and especially for how the light from the particles affected the environment.

The observation and discussion also revealed several problems:

- Using the keyboard to interact with the start menus was worse than anticipated for people new to VR.
- The modeshifts were often not understood.
- When placing the Leap Motion on the table many users required guidance to find it.
- The fireworks exploded a bit too high up.
- Tracking reliability was a huge problem and confused many users.

6.2 Questionnaire

To get more opinions on the value of using the application in VR compared to on a screen and to get better documentation of how easy it was to understand and use the gestures a test with more focused questions were performed. 10 students from a class in "Virtual Reality in Theory and Practice" participated in the test. With the interest the testers has in the subject a questionnaire can be used since they are likely to be motivated enough to write down full answers to open ended questions [28].

The tests of Control Sphere included the VR version, with the Leap Motion placed on the table, and the monitor version. To avoid affecting the result of the test due to the order the applications were tested half of the testers started with the VR version and the other half with the monitor version.

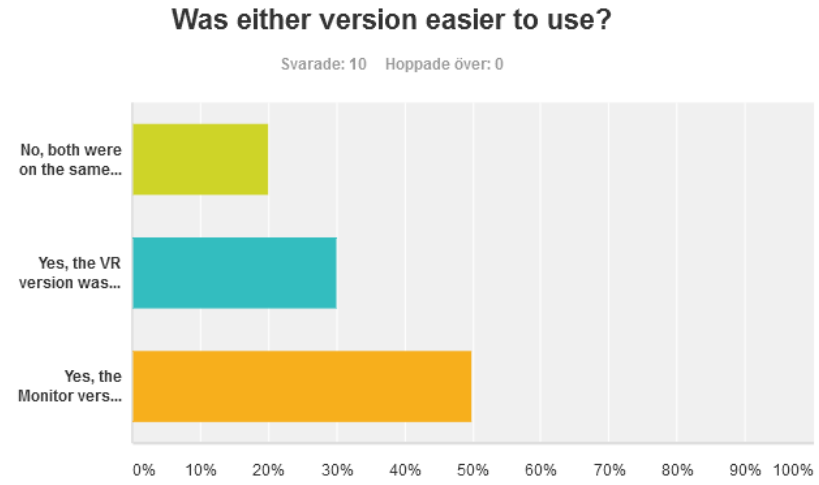
The testers were informed that the Leap Motion tracked their hands and that when they were done there would be a questionnaire to fill out. Since I had already made conclusions from observation tests regarding problems with navigating the start menu, I attempted to minimize it's effect on the user experience by controlling the keyboard on my own in place of the tester.

To avoid confusion in interpreting the answer the questionnaire gave the testers options on their opinion, for these questions there is an additional option to not agree with either opinion. To better understand the reasoning behind the answer and get more qualitative data these clear answers has an open ended follow up question. For questions less distinct answers, such as how easy it was to understand and use a gesture, the testers give the answer by giving a rating on a Likert scale [28].

The following questions were asked:

- Was either version easier to use?
  - If it was please describe how.
- Did either version feel more natural to use than the other?
  - If it was please describe how.
- Was it easy to control the particle color?
- Was it easy to control the particle shape?
- Was it easy to control the number of particles?
- Do you have any further comments on the user experience?

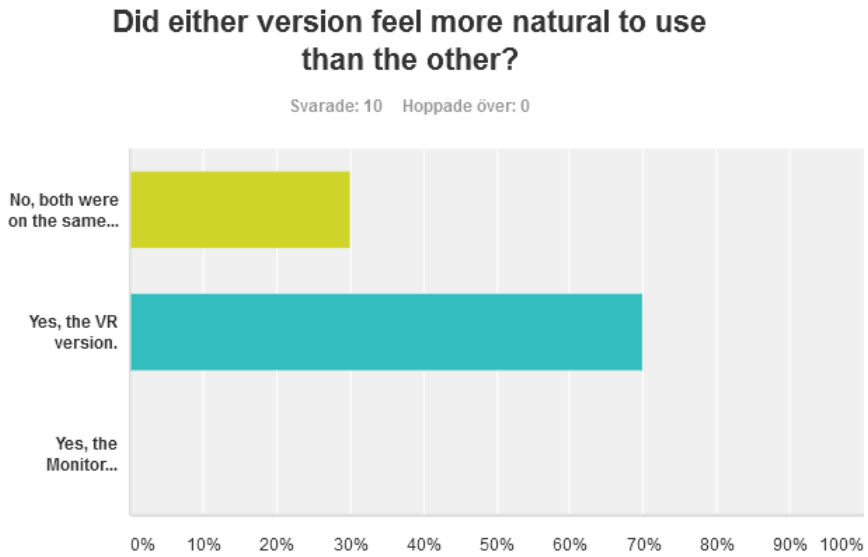
6.2.1 Results



Svarsval	Svar	
No, both were on the same level.	20,00%	2
Yes, the VR version was easier.	30,00%	3
Yes, the Monitor version was easier.	50,00%	5
Totalt		10

If it was please describe how.

- Yes, the monitor version was easier.
  - But only slightly, because I could see where to put my hands
  - It was easier to see where the hand motion device were located so i knew where to put my hands. Otherwise it was the same
  - It was easier to know where to put your hands in front of the sensor when not using the VR version.
  - It was easier to see the ”interactable” area.
- Yes, the VR version was easier.
  - You saw the hands more clearly
  - The sphere felt closer to me and easier to interact with. The negative aspect was that it was harder to get the hands in the right place to get the sphere.
  - It was nice to be able to look around.



Svarsval	Svar	
No, both were on the same level.	30,00%	3
Yes, the VR version.	70,00%	7
Yes, the Monitor version.	0,00%	0
Totalt		10

If it was please describe how.

- Yes, the VR version.
  - It felt like I was inside this fantasy world, so controlling with my hands in the air felt more appropriate.
  - Yes the VR gave a more rich experience. The fireworks had a better feel to it because you had to look up when viewing them like in reality.
  - When looking around in the space with the VR the head tracking made the environment much more natural, even if the environment did not have the best resolution.
  - You felt more involved than in the monitor version.
  - Presence was MUCH higher.

Was it easy to control the particle color?

Svarade: 10 Hoppade över: 0

	I didn't realise it was possible.	(ingen etikett)	It was tricky to get right.	(ingen etikett)	Very easy.	Totalt	Viktat genomsnitt
(ingen etikett)	10,00% 1	0,00% 0	40,00% 4	40,00% 4	10,00% 1	10	3,40

Was it easy to control the particle shape?

Svarade: 10 Hoppade över: 0

	I didn't realise it was possible.	(ingen etikett)	It was tricky to get right.	(ingen etikett)	Very easy.	Totalt	Viktat genomsnitt
(ingen etikett)	30,00% 3	0,00% 0	50,00% 5	10,00% 1	10,00% 1	10	2,70

Was it easy to control the number of particles?

Svarade: 10 Hoppade över: 0

	I didn't realise it was possible.	(ingen etikett)	It was tricky to get right.	(ingen etikett)	Very easy.	Totalt	Viktat genomsnitt
(ingen etikett)	20,00% 2	10,00% 1	30,00% 3	20,00% 2	20,00% 2	10	3,10

Do you have any further comments on the user experience?

- "Great user experience, taking it further would maybe be some way to move around in the world."
- "Just because the hand tracking device wasn't that good, the VR experience didn't feel so strong."
- "I liked the experience, Did not fully understand the small spheres in the outline of the big white one. The VR experience was also enhanced by the music. It took some adjustment to get the desired color I wanted. There was a learning curve to understand the "behaviour pattern". When using the VR I had already practiced on the "normal" version."
- "It was a lot of fun! Really cool."
- "When using Oculus it would be nice to have direct feedback of where the culling of the leap controller is."

6.2.2 Interpretation of Results

To get a clear analysis of the gathered data the answers to each question is reviewed.

- Was either version easier to use?
  - The testers seem to agree that being able to see the Leap Motion made it easier to perform the interactions. Since a large number of user seemed to prefer the feeling of the VR version I believe the improved user experience is the reason for people finding the VR version easier to use.
- Did either version feel more natural to use than the other?
  - Despite half of the users finding it easier to use the monitor version no one thought the monitor version felt more natural. Every comment from the user strengthens my own belief of the value added by placing the gesture interface in VR. The comments suggest that the environment and effects improved the feeling of the interaction.
- Was it easy to control the particle color?
  - All but one person seems to have made the connection between the color of the fingers and the color of the particle. Most users seems to have found every interaction quite tricky. I believe this is due to unreliable tracking and some trouble finding the tracking volume in the VR version.
- Was it easy to control the particle shape?

  - Too many testers failed to realise the connection between the particle shape and the mode shifts. During early development the mode shifts seemed to be understood in every user test. I believe the particle in the middle to be a bit to out of focus and obfuscated by the spinning pentagram. The connection between the glowing spheres and the selected particle also needs to be made more obvious to the user.

- Was it easy to control the number of particles?

  - This result was a bit of a surprise. The number of particles can be controlled by triggering the firework before all of the points of the pentagram are occupied. I believed some users would not realise this was possible which would be alright since it is not an important aspect of the interface. But I expected the testers who understood it to find it easy to control since squeezing the sphere to activate the firework is a gesture no tester have had any trouble to perform. The trickier tap gesture for adding of a particle can be interpreted as a part of the control of the number of particles. The question should have been stated differently to remove any such ambiguity.

- Do you have any further comments on the user experience?

  - The testers generally seem to have had a good experience with Control Sphere.

    The learning curve mentioned by one of the testers is a great sign that the feedback on the actions work well, and helps the user understand the interface in the way I intended.

    Moving around in the world was removed from development early on due to challenges in locomotion and the focus on the gestures. It would have been nice to have but instead I surrounded the player with a flowing river to make the constraint natural.

    Direct feedback of the culling of the leap controller was something I experimented with early on with the placement of the Leap Motion on the table but there is no way of knowing where the tracker is physically positioned in relation to the user's head. The solution to this is to mount the Leap Motion on the Oculus Rift.

    Reliable tracking would make every gesture easier to perform for the user. The user would then get feedback on the correct action and could easier understand the gestures. The comment on unreliable hand tracking being detrimental the the VR experience is collaborated by the research suggesting mismatches in animation can cause nausea discussed in chapter 2.1. In my opinion it is very clear that reliable tracking is absolutely vital for a good user experience of gesture based interfaces and even more so in VR.

Chapter 7

Updating Control Sphere

After the observation tests and the questionnaire result I revisited the design. The updated design can be seen in figure 7.1. This version has not been extensively tested.

To begin with I updated the project from Unity 4 to Unity 5 as I no longer had the pro version of Unity four and the features I needed from the pro version were free in Unity 5. As I updated the project I got the latest version of the plugins from Oculus and Leap Motion and got everything working.

I made some small tweaks to make the selection of a particle be easier to notice and better tied to the modeshift signifiers. Some additional particles were added that shows the particle you will get before you do a modeshift. This overview was something I have wanted since I first evaluated the 3D Sphere application. These new objects are an additional concept for new users to grasp but for me it was unquestionably an improvement as I didn't have to remember the hand positions for the different particle shapes. These four new elements also gets updated when a modeshift happens to show the new possible options. The option to place the Leap Motion on the table was removed. The program settings screen was also removed from the updated design and so at startup you get a prompt to hit space to recenter the Oculus and are then taken to the tutorial video. I did not spend time on updating the tutorial video, so it shows the original design. To remove the tutorial video and begin the experience I required users to use the activation gesture. A screen based version of the updated design was also implemented.

7.1 Description of Application

The setup process was simplified somewhat. From the first scene in the startup menu the option to select placement for the Leap Motion was removed. Only the prompt to recenter the Oculus orientation was left. The options menu was removed altogether so after hitting space once the user was transitioned to the tutorial video. Below the video the information was changed to prompt the user to do the activation gesture. Pressing space again would recenter the Oculus again so the user is required to use the gesture. It is also possible to hit escape and bring up the credits.

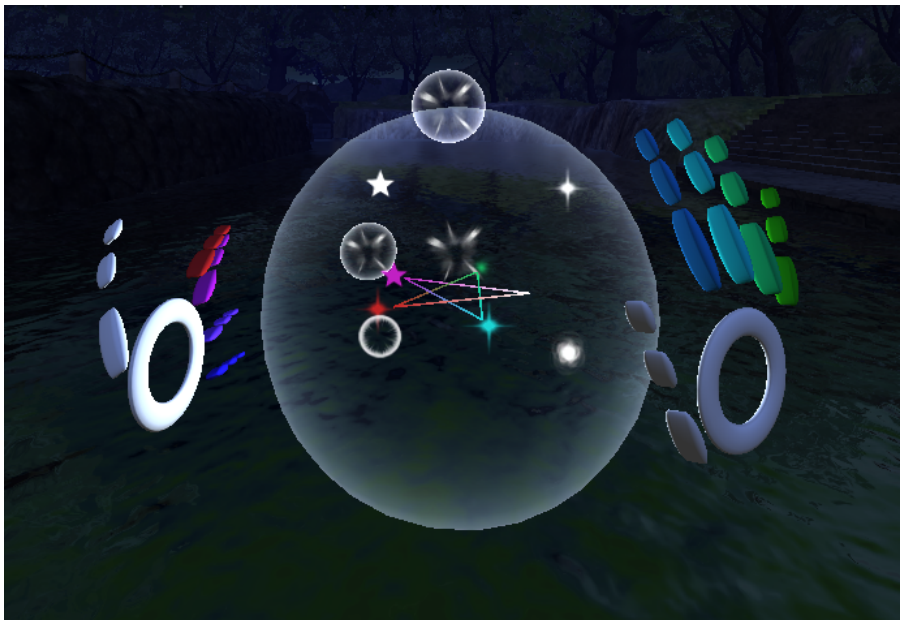


Figure 7.1: The Updated Design

The design has also been updated, as seen in the figure 7.1. The center sphere is removed and the center particle is moved up. The animations that previously targeted or originated from the particle still does and uses the updated location. The modeshift signifiers now contain a copy of the center particle and there are four new particles displayed that the modeshift signifiers move through during a modeshift. When this happens the modeshift particle takes on the shape of the particle it passes through and the four new particles changes shape. The center particle changes when the modeshift is done.

The point where the trail effect stops and the firework explodes is lower in the new version.

Chapter 8

Discussion and Conclusions

This final chapter discusses the design challenges and solutions found and applied during the project. The decision to diverge from VR Typing to Control Sphere is also explained. Further I discuss my conclusions from the project about using gesture interaction in VR in general and with the Leap Motion specifically. Finally the conclusion of the thesis considers how well the Control Sphere application managed to fulfill the goal of the thesis.

8.1 VR Typing

I spent a lot of time on developing VR Typing and I had intended for it to be the only project for the thesis. I had done a lot of testing of the design during development, however it was only at the end of the implementation that I got any output to work and was able to get a feeling of what it was like to use the application for its intended purpose. With a few observational tests I evaluated how well the implemented features conveyed the possible interactions with the interface. The positioning of the text inside the sphere with the added color coding seemed to work well as testers did not ask about it and understood how what finger triggered what command. The sound feedback and color flash on a successful tap was also a clear indicator of a successfully performed action. The ghost hand with the sequence of steps that they emulated and had to be followed to remove them was not understood at all. They were often ignored and left simulating as the user explored other aspects of the interface. The light up menu was also ignored as testers focused on the sphere when interacting with it.

The reason I decided on a completely different design and not improving on these features was however based on my personal experience of frustration when writing sentences using VR Typing. VR Typing has issues with reliability described in depth in chapter 8.6.3. This means that some input won't register and some unintended input will. When using the application for typing the user has a clear plan of the intended output of the system. Any mistakes from the system will break the stream of commands the user intended and the user has to form a new plan and spend time to correct the error. The user can relate this to familiar reliable systems, such as keyboards and touch screens, making VR typing

look worse by comparison. Even if the lack of overview of the keys and the lackluster environment was addressed it would still be a difficult process prone to errors to type simple sentences.

8.2 Placement of Information

While the sphere design has good affordance as an object to hold and touch, it makes positioning information problematic. As the sphere is always in close proximity to the hands, the hands will often occlude information about the sphere.

The "Light Up Menu" was an attempt to workaround this by placing information about the sphere somewhere else. This proved ineffective as the user was focused on the sphere when the information provided by the menu was useful.

Since the user is focused on their hands and the sphere, the information about the interface needs to be placed within this interaction space. To move information about the interface it is possible to move the interaction space. This could be done activating different features of the sphere by making it touch an object or be in a specific area. In the current implementation of Control Sphere the positioning does not affect the interaction. By giving it an effect the design would make use of the spacial property of VR.

8.3 Directing the User's Focus

When interacting the user is focused on the sphere. This is where the detected hands are and where all the animation is happening. But in Control Sphere I want to direct the user's focus from the sphere to the point where the firework appears. I went to great lengths to achieve this. The gesture to trigger the firework is distinct and a natural stopping point in the interaction. With a flashing particle effect and a distinct sound the sphere is deactivated for further input with a trail going from the sphere. The trail starts by going forward to allow the user time to focus on it before going up to the point of explosion leaving the user anticipating a conclusion to the effect.

In testing this worked very well and in the cases where the first explosion was not in focus it drew enough attention with the sound light and particles emitting from a single point for the user to make the connection and be ready the second time.

8.4 Information Noise

Early on in the Control Sphere design observation tests gave the impression that the users understood the connection between the movement of the glowing orbs and the particle change quite well. However the questionnaire results showed that this was not the case. My interpretation is that the rotating pentagram and small sphere surrounding the center obfuscated the connection and distracted the user with unrelated information noise.

I attempted to solve this in the updated version by distancing the center particle from the pentagram and making the connection clearer by changing the glowing orbs to also contain the selected particle.

8.5 Understanding Control Sphere

8.5.1 Introduction Video

The introduction video shows the users the possible gestures with a text explaining what each gesture does. After watching it the user is free to experiment with the interface.

As several users did not understand all gestures and their effect, the video was not a sufficient tool for learning. I find it likely that several users did not read the text in the video. What the video did succeed in was to give the user and example of the possible interactions. After the video every user fired a combination of particles without further instructions. This was probably possible due to the limited number of gestures required and the user having seen how to perform them once.

8.5.2 Interaction Loops

In chapter 2.3, Human Computer Interaction, we describe the communication between the user and the system with a feedback loop of four parts.

- The user's mental model of the system
- An action on the system taken by the user
- The rules applied on the action by the system
- Feedback to the user from the system

All of the interaction loops of Control Sphere were identified and the design was adjusted to give every possible interaction good hints and affordances to inform the user's mental model and clear feedback on the actions effect on the system. This allows the conversation between the user and the system to flow and using the system becomes a learning experience for the user.

In Control Sphere interaction loops as defined by the actions of the user are:

- Moving the hands and the sphere
- Mode shifts
- Taps
- Adding the last particle
- Squeezing the sphere

The feedback of every action was fine-tuned to give clear information and to give the user the best possible experience. For example the positioning of the sphere which is a continuous action was given a lot of focus during the development of VR Typing. For the other gestures there is a distinct sound combined with a number of animations flowing into each other.

The effect of the small details in feedback can be hard to identify. One example was during the development when observation tests were performed with some missing animations. When animations were later added a user who had tested it without them did not realize what had changed but remarked that the sphere felt better to use.

During the animation the user gets a moments pause to observe how the system reacts. As every added particle emerges from the selected particle with a color matching the tapped finger, the user can come to understand the connection over time. When the last particle is added, the tap and modeshift are disabled to show the user that another action is required. For this more conclusive gesture of squeezing the sphere, the feedback is made more rewarding in the form of the complete firework.

A problem I had in several loops was applying the correct rules to the user's action due to failing tracking. This often broke down the conversation with the user and created confusion.

8.5.3 Uninterrupted Interaction

To decrease the impact of failing tracking Control Sphere was designed with output that had a smaller effect on the goal of the user than the intended sequence of letters of VR Typing.

When failing tracking does not affect the next action of the user interaction can continue as planned instead of forcing the user to form a new plan and spend time to correct the error. As the user is not interrupted it becomes easier to enjoy the experience of interacting with the sphere even when tracking sometimes failed. When the user gets comfortable with the interaction they can achieve a sense of proficiency.

8.6 Leap Motion in Virtual Reality

8.6.1 Placement of the Leap Motion

During the questionnaire in chapter 6.2 the placement of the Leap Motion was on the table. This was also the default placement in the application with the option to switch to the headmount. The reason for this was that I experienced better tracking reliability on the table. I had also gotten very used to finding the tracking space on the table during my months of development without support for the headmount. For the testing performed during development I had guided the user to begin the experience, in doing so I failed to test the setup process.

As mentioned by testers in the questionnaire, the HMD version of the application is harder to use due to how difficult it is to find the tracking volume of the Leap Motion on the table. Orienting the Leap Motion on the table and resetting the tracking volume also makes for a very difficult setup process for first time users.

For VR it is important to calibrate the interface so that the virtual and physical space match. By placing the Leap Motion on the table it's calibration is estimated and it can be difficult for the user to know the positioning of the tracking space. This information is vital for interaction. I believe this was the reason many testers thought the screen based version of the application easier to use. By placing the Leap Motion on the HMD this problem is solved.

8.6.2 Gaze Directed Tracking Volume

With the Leap Motion mounted on the HMD the tracking volume follows the gaze of the user. As the user is required to perform gestures within clear view of the camera, the interaction can be performed only by looking down at the hands or by putting the hands up where the user is looking.

Holding up the hands is very tiring when interacting for extended periods of time and if the user should be looking down the interesting and relevant information about the world should be placed there. This is a poor scenario for any continuous interaction in VR as the virtual environment surrounding the user should be interesting to look at.

With regards to this aspect Control Sphere works fairly well. The distinct breaks in the interaction where the users focus is redirected to the scenery becomes more valuable.

Other discrete interactions, such as reaching out to touch a virtual object the user is looking at also works well.

8.6.3 Tracking Reliability

The main problem with both VR Typing and Control Sphere is the reliability of the tracking.

Tracking reliability is a very big issue since failing tracking breaks the feedback loop of the system. As the user attempts to understand how to interact with the system it is vital that the system can interpret the actions correctly and provide the appropriate feedback. Any incorrect tracking of input does not only give incorrect output, it also gives the user incorrect information about how the system should be used and how it is intended to respond to an action. These are fundamental challenges for the user in any Human Computer Interaction and failing tracking is an additional source of confusion.

In combination with VR unreliable tracking can break the user's sense of presence when the virtual representation of a hand suddenly behaves differently from the actual hand. This had a very negative impact on the Control Sphere application. When commenting on the experience one of the testers focused on this problem and stated that: "Just because the hand tracking device wasn't that good, the VR experience didn't feel so strong."

As a developer using the Leap Motion hardware and software you are unfortunately limited in the ways you can address the tracking reliability. The main thing that can be done is to use gestures and hand positions that can be easily detectable and testing this early in development [26].

One technique I implemented slightly improved detection of what finger was used for the tapping gesture. With the Leap Motion placed on the table and the hand positioned in a plane orthogonal to the cameras only the bottom finger is seen clearly. So sometimes the tracking assumes that the fingers above follows if this finger bends. So by iterating over the fingers from bottom to top and only detecting taps with one finger the correct finger is detected. However with the camera placed on the HMD the topmost finger is what's seen and so the fingers are iterated over from top to bottom in this case.

Another good example is the Erghis development. Erghis began with the concept of the sphere, a focus point of versatile and comfortable small gestures. During the development of Erghis First for Leap Motion the tracking was very poor. To achieve better tracking the gestures had to be adapted to make the hands face the camera. The interface of Erghis First showed this hand position for the user to mimic. When the Erghis 3D Sphere was developed the tracking of the Leap Motion had improved significantly and the gestures returned to be centered around the sphere. This gave a better model and focus for the gestures at the cost of some tracking reliability.

The reason that the change of position from the table to the HMD caused a downgrade in tracking reliability is that the hand position and gestures from Erghis are not adapted to this direction of the tracking volume. With the current positioning of the sphere, attempting to improve tracking by facing the palms towards the HMD will place the sphere right in the user's face. It took time to come to this harsh realization about the viability of using Erghis with the Leap Motion in VR.

8.7 Gesture-Based Interaction in Virtual Reality

8.7.1 The Challenges

A major part of the challenge is to create visuals where every response meets the user's expectation. Gesture based interfaces are called natural user interfaces and if the user can't understand the way the interface responds or if there is a mismatch between the user's prediction and the actual result it is not a natural feeling. Managing the user's expectation on an interaction is much easier when the interaction is kept simple. The challenge is then to design complex systems to be controlled with simple interactions.

It is also difficult to avoid unintended input for gesture based systems. Part of this challenge is to design a system that the user can't unintentionally interact with by just being present in the tracking space. This means avoiding gestures that can be performed accidentally.

The gestures used needs to be reliably tracked for the interface to be reliable. To design reliable gestures the limitations of the tracking system has to be taken into account. For

camera based interfaces this means designing gestures the camera can easily see. The visuals of the system should give the users cues to position their hands in a way that can easily bee seen.

For the Leap Motion the camera should be attached to the HMD to use with VR. This means looking at the place gestures are performed. If the interface is intended for continued use, designing for comfort can be a challenge. The user should not be required to hold up their hands at eye level too much and should at the same time not be constrained to look down, not being allowed to look around the virtual world.

Another challenge is efficient placement of information. As the Erghis interface follows the hands and the hands can only be tracked when looking at them, important information about the interface has to be centered around the hands. It has to be taken into account that the hands can obstruct information.

With all the challenges in mind it is very difficult to design gestures for a system like the Leap Motion to be an an efficient core mechanic for manipulating virtual worlds for extended periods of time.

8.7.2 The Value

When comparing the use of gestures in the monitor version and the VR version of Control Sphere in the questionnaire 7/10 testers agreed that the VR version felt more natural with no one thinking the opposite. This is despite many thinking that the VR version was harder to use.

This natural feeling likely comes from viewing the virtual hands in the position the sense of proprioception tells the user feel that their real hands are. As the hands move as expected in relation to the rest of the virtual environment the user has the ability to reach out and touch virtual objects. That anyone using this combination of technologies can manipulate the orientation and position of two virtual objects simultaneously is an extremely powerful and engaging method of interaction.

As the limitations existing in current hardware and software disappears, gestures will become a viable method for any interaction in VR. Until then there are many design experiments to perform with what we have today.

8.8 Conclusions

The goal of the thesis was to use the potential benefits of combining VR and gesture detection to introduce new users to the Erghis Sphere gestures.

The resulting application that somewhat achieved this is Control Sphere.

In the Control Sphere application the benefits of of VR was diminished due to the changed position of the Leap Motion from the table to the HMD having a negative impact on the tracking reliability for the gestures. The questionnaire also shows that 3 out of 10 testers did not understand important parts of the interaction after about 5 minutes of use, making the application lacking as an introduction.

However there has been more than 200 downloads of the application with very positive feedback from testing. Testers said the application provided a "Great user experience" and the gestures were experienced as an engaging method of interaction. A version of the application was created for a regular monitor and user tests confirmed that performing the gestures in VR was a more natural experience even though the monitor version was easier to use.

The application was improved in the updated version to make it easier to understand the gestures that were difficult to understand and to provide a better overview of the options of the system. The introduction to the application was also improved by simplifying the start menu and integrating the activation gesture to start the experience after the introduction video.

The flow of the interaction worked well in Control Sphere, when interacting the user can focus on the sphere and it's visual feedback and the natural pauses in the interaction gives the user opportunity take in the surrounding virtual environment as the gaze of the user can follow the trail of the firework into the scene.

The gestures used could be the main method of interaction in many different VR experiences as they are a very natural and engaging method of interaction however for the communication between the user and the system to work as intended the gestures needs to be reliably tracked.

Bibliography

[1] Erghis Technologies AB. *Erghis*. URL: http://erghis.com/.

[2] Steve Bryson. *Virtual Reality: Definition and Requirements*. URL: http://www.nas.nasa.gov/Software/VWT/vr.html.

[3] Martijn J. Schumie et al. *Research on Presence in Virtual Reality: A Survey*. URL: http://graphics.tudelft.nl/~vrphobia/surveypub.pdf.

[4] Tom Forsyth. *Connect: Developing VR Experiences with the Oculus Rift*. URL: https://www.youtube.com/watch?v=addUnJpjjv4.

[5] Eugenia M. Kolasinski. *Simulator Sickness in Virtual Environments*. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA295861.

[6] Oculus. *Oculus Best Practices*. URL: http://static.oculus.com/documentation/pdfs/intro-vr/latest/bp.pdf.

[7] Richard Yao. *Oculus Connect: The Human Visual System and the Rift*. URL: https://www.youtube.com/watch?v=6DgfiDEqfaY.

[8] Mikael Abrash. *When it comes to resolution, it's all relative*. URL: http://blogs.valvesoftware.com/abrash/when-it-comes-to-resolution-its-all-relative/.

[9] Dale Purves, George Augustine, and David Fitzpatrick et al. *Neuroscience. 2nd edition*. Sinauer Associates, 2001.

[10] Mikael Abrash. *Down the VR rabbit hole: Fixing judder*. URL: http://blogs.valvesoftware.com/abrash/down-the-vr-rabbit-hole-fixing-judder/.

[11] John Carmack. *Latency Mitigation Strategies*. URL: https://web.archive.org/web/20140719085135/http://www.altdev.co/2013/02/22/latency-mitigation-strategies/.

[12] Oliver Kreylos. *Fighting black smear*. URL: http://doc-ok.org/?p=1082.

[13] HTC. *htc Vive*. URL: http://www.htcvr.com/.

[14] Oculus. *Introducing Oculus Touch*. URL: https://www.oculus.com/en-us/rift/#oculus-touch.

[15] Sony. *PlayStation VR*. URL: https://www.playstation.com/en-au/explore/ps4/features/playstation-vr/.

[16] Leap Motion. *Leap Motion Controller*. URL: https://www.leapmotion.com/.

[17] Donald norman. *The Design of Everyday Things*. Basic Books, 1988.

[18] Paul Pangaro Hugh Dubberly Usman Haque. *What is Interaction*. URL: http://www.dubberly.com/articles/what-is-interaction.html.

[19] Daniel Cook. *Loops and Arcs*. URL: http://www.lostgarden.com/2012/04/loops-and-arcs.html.

[20] Daniel Plemmons and Paul Mandel. *VR Best Practices Guidelines*. URL: https://developer.leapmotion.com/vr-best-practices.

[21] Alex Colgan. *Designing VR Tools: The Good, the Bad, and the Ugly*. URL: http://blog.leapmotion.com/designing-vr-tools-good-bad-ugly/.

[22] Barrett Ens and Rory Finnegan Pourang Irani. *The Personal Cockpit: A Spatial Interface for Effective Task Switching on Head-Worn Displays*. URL: http://hci.cs.umanitoba.ca/assets/publication_files/PersonalCockpit_CHI14_Ens.pdf.

[23] Mike Alger. *Visual Design Methods for Virtual Reality*. URL: http://aperturesciencellc.com/vr/VisualDesignMethodsforVR_MikeAlger.pdf.

[24] Oliver Kreylos. *2D Desktop Embedding via VNC*. URL: http://doc-ok.org/?p=824.

[25] Randy Pausch Richard Stoakley Matthew Conway. *Virtual Reality on a WIM: Interactive Worlds in Miniature*. URL: http://www.cs.cmu.edu/~stage3/publications/95/conferences/chi/paper.html.

[26] Alex Colgan. *6 Principles of Leap Motion Interaction Design*. URL: http://blog.leapmotion.com/6-principles-of-interaction-design/.

[27] Christopher Shields. *2D vs 3D in a touch-free hand gesture based interface: An exploration of how 2D and 3D visual aids affect a user's ability to learn a new interface*. URL: http://lnu.diva-portal.org/smash/record.jsf?pid=diva2:756980&dswid=-5289.

[28] Yvonne Rogers, Helen Sharp, and Jennifer Preece. *Interaction Design beyond human-computer interaction*. John Wiley & Sons Ltd, 2011.

[29] Pirkka Rannikko. *User-Centered Design in Agile Software Development*. URL: http://uta32-kk.lib.helsinki.fi/bitstream/handle/10024/82310/gradu04854.pdf.

[30] Inger Boivie Bengt Goransson Jan Gulliksen. *The Usability Design Process – Integrating User-centered Systems Design in the Software Development Process*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.567.8461&rep=rep1&type=pdf.

[31] Ken Schwaber. *Agile Project Management with Scrum*. Microsoft Press, 2004.

[32] Unity. *Unity - Game Engine*. URL: http://unity3d.com/.

[33] Leap Motion Developer. *Leap Motion 3D Jam: Presented by IndieCade*. URL: http://blog.leapmotion.com/leap-motion-3d-jam-presented-indiecade/.