UNIVERSITY OF HULL

Target Practice & VR Accessibility: Shooting Game In Mixed Reality

Final report

Submitted for BSc in
Computer Science for Games Development

April 2020

By
Olumide Oluwatosin Temitope Osikomaiya

Word count: 9291

UNIVERSITY OF HULL

Project Abstract

The following report discusses the steps taken during development of the honours stage project task which was to develop a shooting game within a method of augmented reality. The research portion of the report briefly touches on the history of modern virtual reality and explains the beginning of Oculus and the Oculus Rift before discussing the current virtual reality market. The research portion of the report also reviews interviews on developing virtual reality games and explains various developmental hurdles and solutions that are unique to the virtual reality platform. The design and development portion of the report talks through the concepts made during the design phase of the project and runs through what was implemented, changed and removed during main programming and the reasons behind the choices made. Finally, the software is evaluated before the project as a collective is evaluated and concluded.

UNIVERSITY OF HULL

Acknowledgements

Thank you to Kevin & Kathryn for their consistent moral support throughout the development of the project, giving me the perseverance and determination to finish it through to the end.

Thank you, the Hull University Gaming Society, for the rallying support and encouragement while running a society and completing an honours project at the same time where others didn't.

Thank you to the Auckland Ledge, my roommates, not only hear me ramble about my programming problems, provide me with the support when my mental health was on the decline and let me use them as virtual reality guinea pigs; but here to remind me when to take a break from working and help relax with Super Smash Bros.

And finally, thank you to Mum. I know you don't understand half of what I am talking about, but you listen anyways and encourage me to keep going when things seem tough.

UNIVERSITY OF HULL

Table of Contents

Project Introduction

This honours stage project is primarily a software development assignment, developing a shooting game within the medium of mixed reality. The assigned project description describes using a method of position and orientation tracking for the player to be able to pick up, hold and fire the gun. Development of the project leaves most factors open from the choice of the headset device to the objective and mechanics of the game so long as the player has targets to shoot as part of the main gameplay.

The project has been structured to present a discussion on the development and production of modern virtual reality titles, displaying and evaluating technologies, techniques and production methods by developers, and completed software to analyse and understand the choices made and the reasoning behind them. Knowledge attained during this research and analysis will be used to produce the software product at the end. One of the key points for this project is accessibility and measures to make virtual reality usable by as many people as possible; with this in mind, the final software developed is aimed at a general audience within the recommended age of most virtual reality devices being 13 years or older.

Aims & Objectives

- To research and learn various techniques of virtual reality development
- To analyse and understand the various constraints and developments hurdles that can occur when working with virtual reality
- To assess and evaluate upon various technologies to determine the most suitable for the software development portion of the project
- Develop various accessibility methods for the project software including options for movement and object handling
- Evaluate development, addressing positives and negatives of the development process and final software to create ideas for potential post-development

Report Overview

The report is structured to be presented in a way that is pseudo-chronological to the progression of the project's active development. This report overview details the various sections of the report and what each section focuses on which can be arranged into five major categories:

- **Literature Review:** The literature review is an analyse and discussion on the research accumulated during the project. The literature review will have a short summary of the road to modern augmented reality as well as explain the current technologies available. This section will also analyse various interviews discussing innovations and developer hurdles using virtual reality. Finally, the literature review will also discuss some current examples of software and games within augmented reality.
- **Project Requirements:** Project requirements lays out the aims of the software development portion of the project. This includes addressing a target audience, addressing what the software will be able to do for the user, marking out the functions that the program will be able to perform and the technical structure it runs on.
- **Software Design:** Software design comprises of a detailed presentation of the project software's architecture and implementation from a pre-development perspective. This section makes uses of various diagrams and class structures to demonstrate code and user interactivity within the software. This section will also be discussing the constraints and limitation constants considered during design.
- **Software Implementation & Testing:** Software Implementation and Testing is a discussion on the main development of the project software. This contains the steps taken to develop

the main and various unique aspects of the software as well as discussing difficulties and design changes made during development and the steps made through the process.

Literature Review

The modern iteration of consumer virtual reality as we know for many people, me included, started with the Oculus Rift Development Kit Version 1 (Rift DK1) releasing in March 29th, 2013. Equipped with only a 7inch LCD screen at 60Hz with a resolution of only 640x800 per eye, the headset today would be considered a cheap and uncomfortable piece of equipment; but during its announcements, demonstrations and eventual release it was a technical revolution. Early entries into virtual reality resulted in bulky, stationery and nausea inducing helmets which were only able to display basic images, resulting in the release of the IIS VFX3D in 2000 being the last the general public saw of virtual reality until the Rift DK1.

Modern VR – First Experiences

From the initial start of the company's Kickstarter, the Rift DK1 was in demand; reportedly raising over $2.4 million and amassing approximately 9,500 total supporters. By the time the company moved into rolling out the DK2 in June of 2015, the company revealed that they sold over 175,000 development kit units to consumers and commercial users alike (Hayden, 2015). Despite a product being in a medium considered extremely niche, it was clear that it was a market truly desired for. When late March 2013 arrived the public finally experienced that desire, albeit very rough.

An article discussing the history of oculus describes the device at release *"as already better than everything that came before it. But it wasn't perfect."* (Kumparak, 2014). The rather clunky and top-heavy design of the headset made it uncomfortable for some users and the low resolution screen combined with the magnification lenses caused multiple issues regarding visual clarity. The headset was prone to two major visual artefact phenomena known as **Ghosting** and the **Screen Door Effect**. Ghosting is the visual appearance of trails behind any moving object on the screen and is usually caused by a screen's slow pixel switch time (Desai, P.R., Desai, P.N., Ajmera, and Mehta, 2014). This effect is only increased by the speed of the objects in the scene and the movement of the user's head. The Screen Door Effect, as suggested by the name, is like looking at a screen door which the display is sat behind (Kumparak, 2014). The empty spaces between pixels, even more so defined due to the lenses' magnification, were very visible when using the DK1 reducing the overall visual quality of the headset.

What can be argued however is the biggest and what is an ongoing hurdle for virtual reality is motion sickness. Motion sickness, dubbed cybersickness when experienced in virtual reality, has debate over how it exactly occurs but it is generally agreed on that when information from different parts of the human body doesn't sync and clashes symptoms of motion sickness can take effect including nausea, disorientation, headaches, sweating and eye-strain (Davis, Nesbitt, Nalivaiko, 2015). Developments have been made in head tracking and more importantly positional tracking in the aim of reducing the chance of motion sickness, the original Rift DK1 however had no method of position tracking. While it can detect a user's head angle, it couldn't detect the user's head in relative space meaning the screen and thus the user's vision stayed static despite head movements – a perfect environment for motion sickness.

These issues were improved upon with continued development. The release of the Rift DK2 added new OLED based screens at full high-definition and positional tracking in the form of an external infrared sensor. This would become the fundamental basis for the future DK2 and the final commercial release of the Oculus Rift.

The Current Market

Throughout the success of Oculus' Kickstarter and eventual commercial release, multiple other companies have been developing virtual reality headsets for the market. The competitive market has given rise to new concepts that have now become common practice resulting in the consumer having a range of options to suit their specifications from available space to motion tracking methods.

Play Area

The space which you use for virtual reality devices is commonly referred to by game developers as the Play Area; there are three commonly described size margins of Play Area and most headsets can perform all three to a varying degree:

- **Seated Play Area:** Limited space environments may not have the capabilities for even limited movement. A Seated Play Area would have the user be essentially stationary, limited to only head and upper-body movement. These setups would require a single facing sensor towards the headset or internal positional tracking.
- **Standing Play Area:** A Standing Play Area doesn't require major movement from the user but would give the freedom for arm movement and extension and very limited positional movement such as rotation and crouching. Positional trackers are required to identify the floor and in turn the current height of the user.
- **Room-Scale Play Area:** A large enough environment (approximate minimum of 2m x 1.5m according to HTC's guide on the VIVE Headset) would be suitable to a Room-Scale experience, allowing large manual movement across the virtual environment – this can provide users with a more immersive experience and arguably can reduce nausea as the user is actively moving their body. Conventional tracking for Room-Scale requires multiple trackers placed around the room to capture as many angles as possible; some headsets however are equipped with their own set of sensor cameras and additional positional trackers allowing for full Room-Scale experiences without additional external trackers.

The Current Headsets

HTC, Valve and Microsoft (through collaborations with additional partners) have developed along with Oculus multiple headset models which make up the current virtual reality top market:

Oculus

- **Oculus Rift CV1:** Released in 2016, the commercial version 1 (simply known as the Oculus Rift) was the first public commercial release from Oculus and the culmination of all the company's development since their initial Kickstarter. The system housed a 2160x1200 OLED screen with a 90Hz refresh rate. The device now supported six degrees of freedom (6DOF) through the addition of the Constellation tracking system, IR sensor towers that were connected to the USB – up to 4 sensors can be connected in a single configuration. These sensor towers also allowed for a standing experience and small scale movement.
- **Oculus Rift S:** Released in 2019 to replace the now discontinued Rift CV1, the Rift S was an overall upgrade with minor sacrifices made to do so. The system housed a larger 2560x1440 screen but was now built with a fast-switch LCD in comparison to the higher quality OLED. The screen also ran at a slightly lower 80Hz refresh rate. The biggest change was that the 6DOF tracking was now done entirely in the headset through 5 built-in cameras; this not only removed the need for external trackers but allowed for a full room scale play area as the user was not bound to the viewing angles of wired sensors.
- **Oculus Quest:** Alongside the release of the Rift S, Oculus also released the Quest. What made the Quest stand out from every other main headset on the market was that it was a

full standalone virtual reality device, removing the need for an additional PC to run the programs. The device ran on a Qualcomm Snapdragon 835 8-core processor with 4GB of RAM, similar to that of high-end smartphones, and was available with either 64GB or 128GB of storage. It housed 2 OLED displays at 1440x1660 running at a 72Hz refresh rate. The quest also had a similar tracking system to the Rift S, using 4 built-in tracking cameras for 6DOF and allowing for a full room scale play area. Later in the life of the Quest through a software update, it was given the ability to be used with a computer as a conventional virtual reality headset.

HTC

- **HTC VIVE:** Released as a collaboration between HTC and Valve in 2016, the HTC VIVE was the first major competitor against Oculus in the virtual reality market. Housing the same screen and refresh rate as the Rift, the unique features came from Valve's side of development through its tracking system and operating platform. Valve as part of development built the OpenVR API and SteamVR platform, allowing virtual reality devices such as headsets and trackers to operate with each other (provided they support the SteamVR platform) and allow developers to easily build software that is compatible with multiple headsets regardless by using OpenVR as an interface. In addition to the SteamVR platform, Valve developed the Lighthouse tracking system; these are external, wide-range and wireless IR sensors that can be placed across a room to provide 6DOF and a full room scale experience.
- **HTC VIVE Pro:** In 2018, HTC released an equipment upgrade to the VIVE known as the Pro. The new device provided a higher resolution screen at 1440x1600 per eye and a more ergonomic design. In addition, the VIVE Pro came with SteamVR 2.0 support and thus support for Lighthouse 2.0; the new lighthouses along with being more precise added support for extra lighthouses to be used in a user's setup allowing for a max tracked space of 10mx10m square size.
- **HTC VIVE Cosmos:** Released in 2019, the VIVE Cosmos was the next evolution for HTC's virtual reality headset. In addition to having built-in tracking through 4 cameras, the screen on the Cosmos was able to be flipped open in order for the user to access the real world without need to fully remove the headset. Additionally, the Cosmos was built with a minor modularity design, an additional face plate can be purchased to add compatibility with SteamVR 2.0 lighthouses to provide the same features seen on the original VIVE and VIVE Pro.

Other Headsets

- **Valve Index:** Valve Index is the companies independent endeavour into building a headset and makes use of SteamVR and the lighthouse technology developed with HTC for the VIVE. As a result, all previous lighthouses and HTC VIVE controllers are compatible with the Index headset. The headset houses two 1440x1600 LCD displays able to run at multiple refresh rates: 80Hz, 90Hz, 120Hz and 144Hz. The headset also makes use of a stereo pair of 960x960 cameras. An additional draw of the Valve Index is the Index controllers which contain a sensor array strap able to read finger position, motion and pressure for more precise representation of the user's hands.
- **Samsung HMD Odyssey (Windows Mixed Reality):** Windows Mixed Reality is a platform built by Microsoft for the Windows 10 operating supporting augmented reality through Microsoft HoloLens and virtual reality through its line of headset in partnership with other companies such as Asus, Acer and HP. The virtual reality headsets also support some software features from the SteamVR platform. The Samsung HMD Odyssey is one of the higher rated headsets making use of a 1440x1600 AMOLED display and 6DOF tracking through two built-in cameras.

- **PlayStation VR:** Developed by Sony for the PlayStation 4 and supported on the PlayStation 5, the PlayStation VR was a method of bringing virtual reality to consumers at a value price as it only required a PlayStation 4 to use – hardware considerably cheaper than a desktop or laptop computer powerful enough to run VR. The system housed a 1920x1080 OLED display, lower resolution than all other headsets discussed, but the screen was able to run at a maximum 120Hz refresh rate with the minimum being 90Hz. The headset also had 6DOF through positional tracking of LED lights present across the headset, these were detected by a single PlayStation Camera placed ahead of the user.

Developing for Modern VR

As the medium is so different compared to creating for a conventional gaming platform, development for games to work in virtual reality is usually done by or collaboration with a dedicated VR team with the steps taken and the questions asked during development differing substantially.

Accessibility and more importantly how much resources towards accessibility is arguably the most important question during virtual reality development. The medium adds a lot more factors to consider as written by an article by Gunheart developer Brain Murphy; factors like room space scalability, movement options and work on nausea reducing effects such as tunnel vision become more important when considering that choices can '*potentially cut your audience in half at the outset…'* (Murphy, B., 2018). It becomes more difficult when factors like movement are direct elements to gameplay, requiring more abstract solutions to attempt to take these accessibility measures without jeopardising gameplay. In Murphy B.'s case with Gunheart, he wanted to make the implementation of teleportation *'into a superpower rather than a compromise'*. Changes made to gameplay included making movement fast and as often to the player as possible and revamping the map design to complement it.

In comparison, director of VR at Sony XDEV Stuart Whyte took a slightly different design approach when developing Blood & Truth for PlaystationVR. In an interview, Whyte acknowledged that the movement in their game can be strenuous and so optimised development around seated play (Wawro, 2019). Focusing a development on a singular and direct method of play removes the need for developing systems for compatibility of other play methods but also adds a limiting constraint when building gameplay concepts, especially with minimal movement in seated play. Whyte explains that the idea of a teleportation system didn't feel right '*didn't seem to fit well in [their] setting*'. The team instead alternatively went for a point-to-move physical system, speed determined by distance (Wawro, 2019).

The niche nature of virtual reality means that it is popular as an attraction activity to partake in. The focus on shorter, more spectator friendly activity changes structure during development. The development team for an Assassin's Creed VR Escape Room was able to build an experience exclusively for a location attraction, which suddenly changes the niche nature of virtual reality into a demonstration for the mainstream. The team decided to work on a multiplayer virtual reality experience built with smaller, puzzle based level design. Due to the nature of the experience being a designed attraction setup, *'it meant [they] could have the same minimum specs for every single partner, so [they] didn't have to spend a lot of time on optimization'* (Gamasutra, 2019). The gameplay of the experience is of an escape room, a type of activity which is approachable to groups of friends or families and is a very passive introduction for virtual reality making it a gentle introduction to virtual reality and generally accessible for most people.

Virtual reality is a sensitive medium and smaller design choices that would conventionally be minor additions or changes become large, gameplay defining decisions as they dramatically effect user experience and the wrong design choice can potentially cause physical issues to the user.

VR Game Analysis

Active play of various virtual reality titles was an important part of the research phase of the project. Experience with current examples and design concepts will help to further develop my design ideas when I start development of the project software – noting down ideas and design choices to take forward and missteps to note down and avoid.

Robo Recall

Robo Recall is an on-rail virtual reality shooter created by Epic Games exclusively for the Oculus Rift and the Oculus platform. The game has you shooting and killing hordes of robots for an arcade style score system. The game makes use of a teleportation movement system and full hand motion tracking in order to pick up various guns and objects. The game recommends a standing play area as it doesn't require much movement but requires dodging bullets by moving within the environment.

The movement in Robo Recall took some initial time to adjust to; it is a two part system in which the user directs the analogue stick to activate the teleportation arc, the direction of the analogue stick determines the new facing position when teleporting and letting go of the stick initiates the action. This system is quite different compared to conventional arc teleportation systems due to the additional layer of facing direction, the use of the analogue direction was initially jarring as attempting to teleport quickly resulted occasionally in misinputs for my desired facing direction, causing mild confusion. Continued used of the system however ended up feeling more intuitive than conventional arc teleportation, the ability to set your facing direction reduces the amount of rotating a player needs to perform lowering the chance to lose a sense of balance or direction. Reducing rotations also becomes useful to avoid a headset's wires becoming tangled with each other.

Access to weaponry in the game is built to be easy and nonstrenuous. Weapons that the player takes to the level are positioned at the hips and just above the shoulders, making weapons easily to pick up when arms are in a relaxed position and upwards bicep bend. Dropped weapons in the environment additionally can be picked up from a distance by pointing towards the UI indicator on the weapons, warping them to the player's hands.

Beat Saber

Beat Saber is a rhythm game developed by indie developer Beat Games for the virtual reality platform – supported by both OculusVR and the SteamVR platform and available on PlayStationVR. The goal of the game is to strike oncoming colour coded boxes in the correct direction to the rhythm of the played song with a pair of Lightsabers. The game requires a standing play area as some stages require crouching and rotation.

Beat Saber is a game with a lot of objects oncoming at the player at high speed and without any measures put in place, can be made difficult to focus on where you need to; Beat Saber addresses this through its method of lighting. The intensity of lighting decreasing as you look away from the centre having the edges of the scenery devoid of raw sources of light; this results in natural tunnel vision and creating a point of focus. This helps to reduce potential nausea while playing and pinpoint the player's gameplay focus.

Thumper: VR Version

Thumper is a rhythm game developed by Drool. The title was initially built for standard monitor play and later was provided support for the OculusVR and SteamVR platforms as well as PlayStationVR. The aim of the game is to survive a visually intense track by performing certain actions to the beat of the music. The game in VR has no motion control implementation, only using the buttons and analogue stick from the gamepad.

Thumper's implementation of virtual reality didn't change how the game's systems or mechanics, it didn't make use of gyroscopic input or 6DOF and was essentially and alternative method of viewing the game. Thumper by nature however is a very visually intense game, the use of virtual reality brought you closer to the visuals and so steps had to be made to reduce the chances of nausea. The camera position and the viewing angle in virtual reality was adjusted so that the swerves caused by bends within the game weren't as intense and many of the particle effects present in the original version have been toned down as to not blind and disorient players and reduce the risk of photosensitive epileptic seizures.

Project Requirements

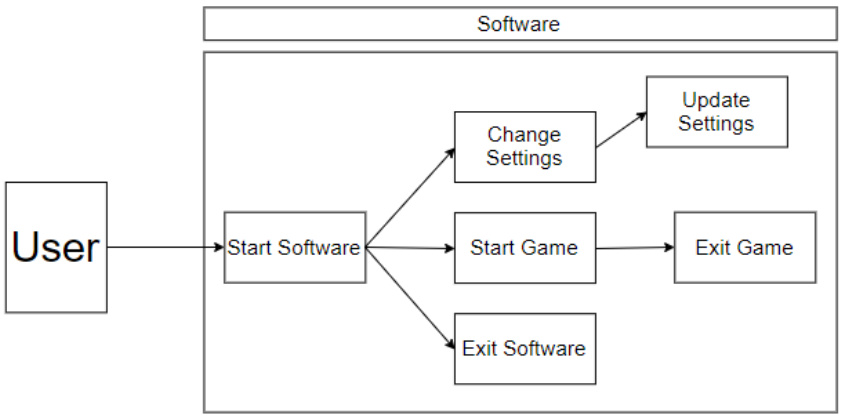
During design and pre-development of the software for this project, a list of requirements and software goals was built to be addressed during main development, testing and evaluation. The list of requirements can be broken down into two main categories: **Product Requirements** discusses the software in relation to potential users such as the target audience and what a user should be able to do while using the software; **Functional Requirements** describes the software in relation to how it is built and how it performs such as use of interfaces, safety and reliability, the desired performance margins and overall quality of the software.

Product Requirements

As the software is being built as a game, the generally considered primary purpose of the software is to provide the users with an interactive environment to complete objectives and some sort of end goal to complete or beat. In addition to this is providing the user with various options within the realm of accessibility in order to have users aim for a more comfortable experience as oppose to not having the accessibility options available to them.

Being developed for virtual reality, the target audience is narrowed by those with the circumstances to own or have access to a virtual reality headset which is a very small market; according to Steam's June 2020 hardware survey, only 1.67% of registered accounts have a virtual reality headset. The age range for the audience varies, Oculus and Samsung have given their headset line a recommendation of ages 13 and up, Sony has provided the PlayStation VR with give the minimum age of 12 years while HTC stated that the VIVE is not designed to be used by children and in fact *"shouldn't be allowed to use the headset at all"* (Gent, 2016). Knowing this, I decided that the minimum age group should be 13 years and up, the advised recommended age provided by the builders of the hardware.

On a surface level, the activity and interactions between states between the user and the software are minimal and if the headset is operating.



Functional Requirements

To meet the laid out expectations and the goal of the project there are core functions that the software needs to be able to perform at the minimum. The core function requirements of the software are listed as followed:

- To be able to trace and track user movement within the environment
- To be able to interact with objects in the environment, such as picking up and throwing objects
- The user will be able to pick up, fire and reload gun objects
- Implement a method of teleportation for the user
- Provide a user with the ability to select and confirm accessibility options provided in the software

Making sure the software is safe to use is especially important for titles that make use of virtual reality. The sensitive medium requires everything to be in smooth operation to avoid any chance at nausea. Steps can be taken in design can be set as requirement constants to lower the chance of these issues, the following which I have decided to take are:

- Avoiding bright flashing lights and imagery in software
- Avoid excessive movement for the user – make the playable software suitable for all play area sizes

The performance of the software is especially important as lacklustre performance can contribute to poor safety and potential nausea due to inconsistent visuals. The software needs to be able to keep a consistently high refresh rate during play, the minimum being 90Hz as that is the most conventional screen refresh rate across headsets. In order to keep to this, I will be focusing my development around a computer system which meets the recommended specs of virtual reality. The specs include a graphics card equivalent to a Nvidia GTX 1060, a processor equivalent to an Intel i5-4590, 8GB of memory and running on the Windows 10 operating system (Lang, 2019).

Software Design

With the requirements set out for the development process, pre-development and the software design starts. While the software design demonstrates the desired finalised concepts, structures and mechanics; they were subject to change during main development.

Design Constraints and Limitations

Understanding the constraint constants that are present during main development provides the ability to lay out what is and is not possible during the development of the software, allowing optimisation of time and resources to create the best product within the given circumstances.

General Constraints

- **Time Constraints:** The project only has a limited time to be completed, software, research and report included with a deadline of April 23rd, 2020. In addition to the deadline, there are other work commitments that I must attend to, further reducing the theoretical time I have to complete the project. Time organisation and allocation is important in order to optimise what short timeframe I have available to me.
- **System Constraints:** While I have a desktop computer system capable of running virtual reality headsets, I don't own a headset myself and as such is limited to what is available to use on university grounds. The university has a large variety of headsets but in limited

quantity, required to be used by multiple other students and lecturers. Development is dependent on the availability of devices on that day.

- **Knowledge Constraints:** Development in virtual reality is completely new to me and such techniques and programming concepts niche to the medium will be foreign to me. The development will be as much learning and familiarising as it is utilising and creating. This constraint compared to others is not really a constraint as I will continue learning during development.

COVID-19 Global Pandemic

In March 2020, the university and all its facilities were shut down for the rest of the academic year due to the currently ongoing COVID-19 global pandemic in an effort to minimise contact between students and staff at the university. This removed my access to a virtual reality device in the middle of main development and as a result stopped it. As compensation, an extension to complete the project was offered which mitigates the time constraints initially discussed. The situation with the pandemic drastically shifted the process for the software development and writing of the report by cutting my development time and shifting my workspace and so a lot of changes were made to the design during implementation.

Hardware Selection

The first choice that needs to be made for the development of the software is what headset main development will be worked on. The use of OpenVR and its compatibility and the availability of the university facilities means I have the choice of all modern headsets on the current market and if needed, changing devices won't cause problems during main development. A note to be made however is that the more premium models such as the Valve Index and the HTC VIVE Pro are in much fewer quantities.

Headset	Screen Resolution	Tracking Method	Optimal Play Area	Platform	Availability
Oculus Rift CV1	2160x1200 OLED @90Hz	Constellation System	Standing	Oculus VR	High
Oculus Rift S	2560x1440 LCD @80Hz	Internal Camera Sensors	Room Scale/Standing	Oculus VR	Low
HTC VIVE Pro	1440x1600 OLED @90Hz	Lighthouse System	Room Scale	Steam VR	High
Valve Index	1440x1600 LCD @144Hz	Lighthouse System	Room Scale	Steam VR	Very Low

Development space is prone to change consistently as I will be sharing spaces with other students except during university after hours; I have to take in consideration that I would have little development space to work with or at least not enough to perform optimal room scale virtual reality. Development at accommodation is not as limited by space compared to university facilities, since the device would be a rental however, it is optimal to select an item that is easy to setup as well as pack-up. The Rift S contains no additional trackers compared to the rest of the devices. As there are multiple headsets available at any time, gaining access to a Rift CV1 would be considerably easier than a Rift S; while the overall quality of the headset is lower, it is still perfectly capable of virtual reality development.

Development Software Selection

Most modern universal game engines are built with support for virtual reality development through OpenVR. There is a choice however to develop an engine using OpenGL and build a framework for virtual reality from scratch, I decided against this as there would be a large amount of development time dedicate to simply implementing the headset as an output and tracking system. My shortlist choice was between Unreal Engine and Unity, two accessible engines that provided a lot of options and built in support for virtual reality which the user can develop on top of. Unreal Engine uses C++ as its programming language, a language which I feel I am less familiar with compared to C#, which is the language used by Unity. For this, I decided Unity would be my main programming engine with use of the OpenVR SDK.

Game Goals & Objectives

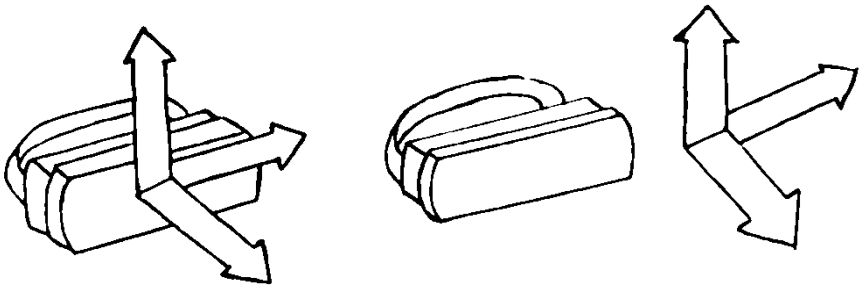
The project task description requires the gameplay of the software involve shooting of targets as part of the objective. The initial idea was to make the game in the style of an obstacle course, having the player make it from the start of the course to the end and having targets shot reducing the time on the stopwatch with the end goal being to gain the fastest time. The course would have consisted of small teleportation challenges and require aiming targets at unorthodox spots, demonstrating the implemented accessibility features, motion tracking and aiming possible within the developed software. The game's overall scope due to the pandemic had to be scaled back considerably and as such the level design of the course was scaled back. The new concept for implementation would be for a linear course with platforms at different elevations, allowing to still demonstrate some of the accessibility features – targets would still able to be short in order to gain a faster time.

Movement System

The design for the player's general movement is split into two main parts: **Grounded Movement** and **Teleportation System**. Grounded Movement focuses on directional movement along the plane, similar to controlling an entity in a more traditional video game medium; Teleportation System is the method the player will be able to traverse gaps or long distances.

Grounded Movement: Head Movement Dependency

Head Movement Dependency is an orientation method for movement within virtual reality. When using this movement, the compass for analogue movement is tied to the orientation of the headset having north always directed as the front of the headset. In scenario, if the player was to hold forward, they would start to turn in world space if they rotated their head. The alternative is having the analogue movement compass tied to world space, requiring manual rotation from another input source.

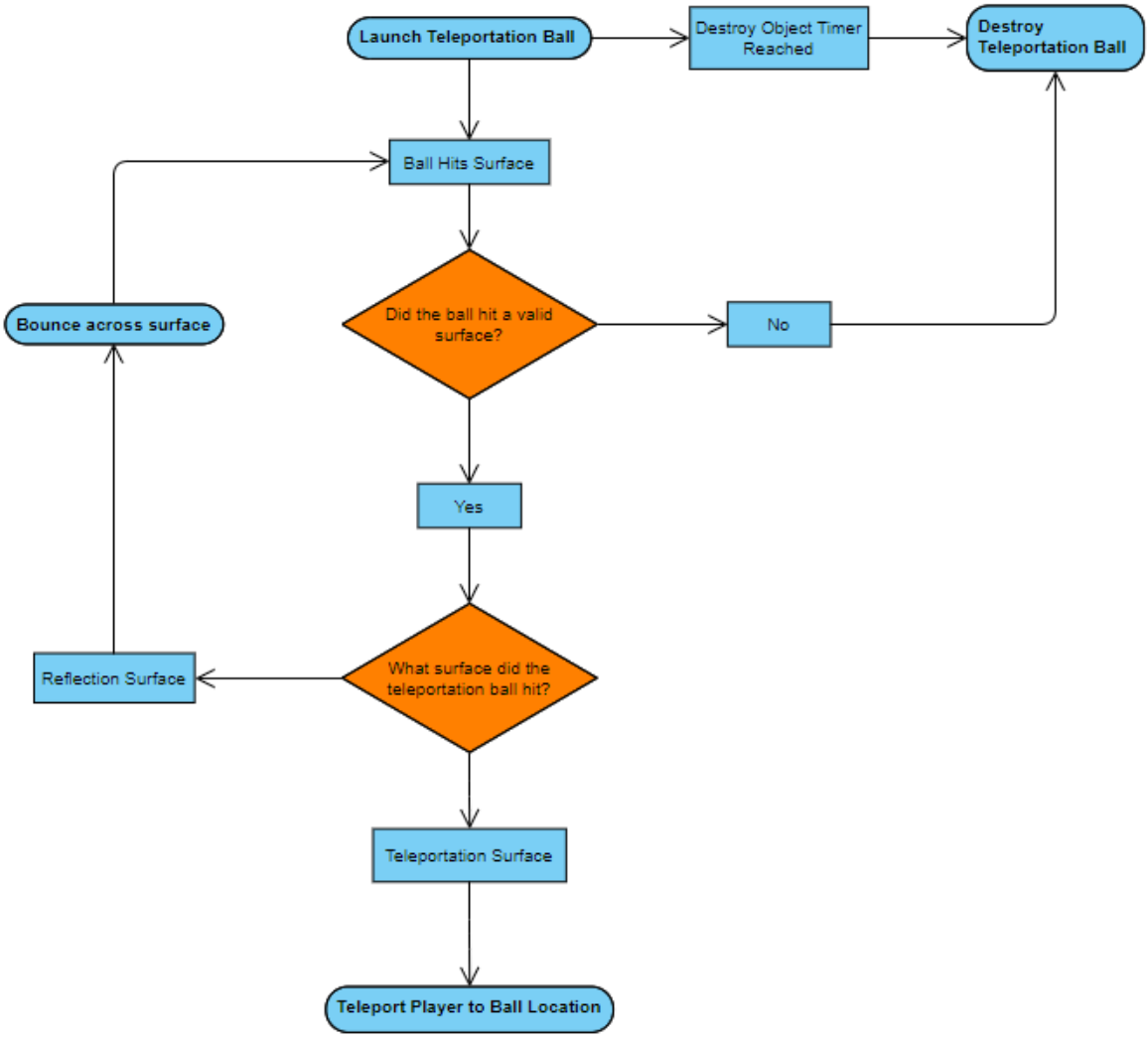


The use of these movement methods is built primarily on preference, some users find the use of head rotation more natural and less nauseating as they are performing a physical reaction that reflects visuals in the environment; others prefer manual rotation for a larger sense of control. As part of adding options for accessibility, both movement methods would be available for selection by the user.

Teleportation System

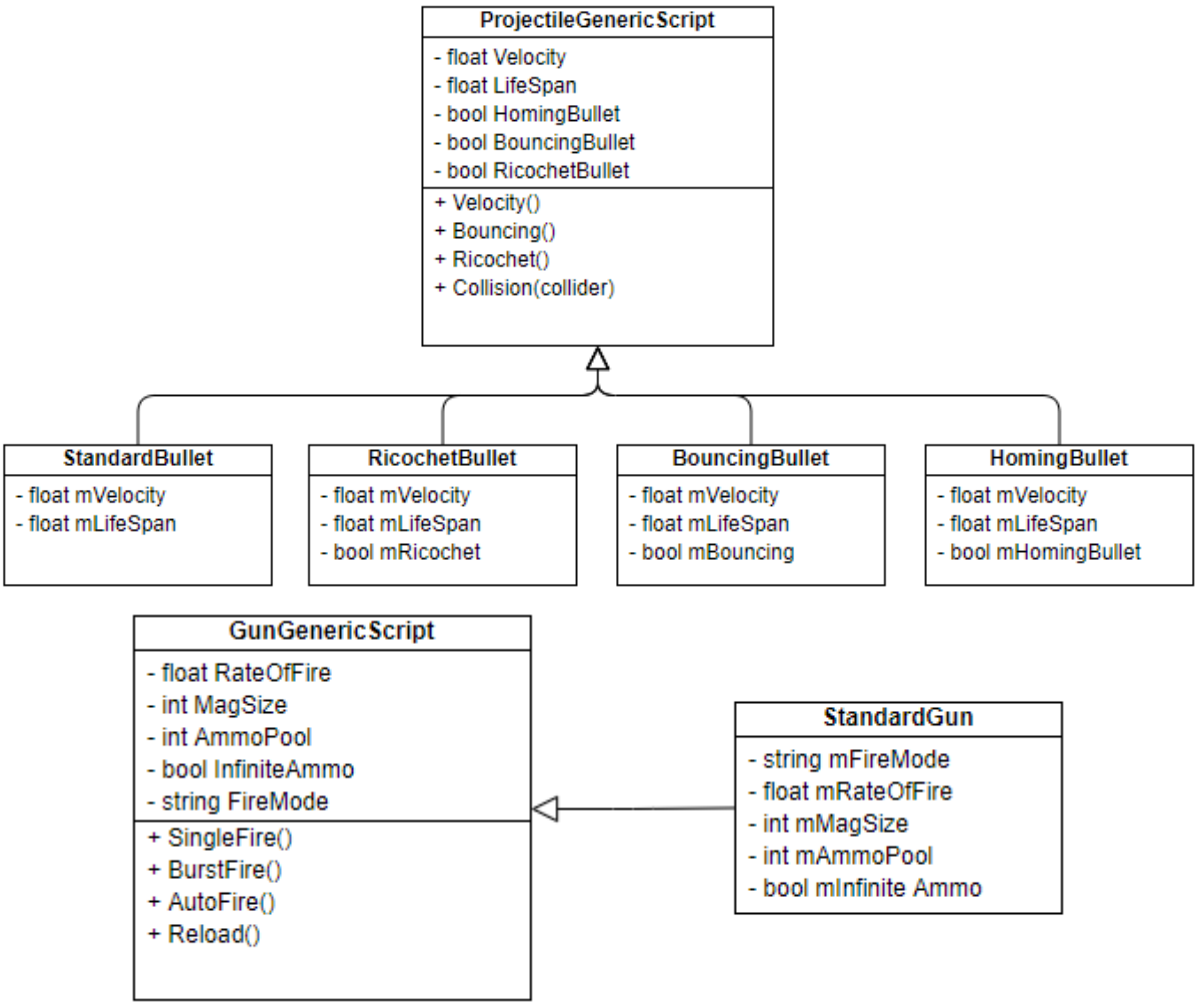
Teleportation within virtual reality game has hit an established standard with arc teleportation. Arc teleportation works and is a very simple to grasp method of teleportation, however for the game I wanted to create a different system that provides a similar result with some more dynamic interactivity compared to arc teleportation.

What was designed is dubbed **Teleportation Ball**. Players would be able to spawn and launch a ball from their hands and when the ball makes contact, the user will teleport to that location. As the ball is launched from the hand's forward direction, the angles accessible for teleportation are essentially infinite. Additional features include having the ability to bounce on special surfaces, extending the potential areas where the teleport ball can reach.



Gun and Projectile Generics & Data Structure

With the project statement discussing that shooting target is to be a main gameplay mechanic within the software, the concept of having a variety of guns in the game has been a set idea from the start of initial brainstorming. While having different features and values, the gun objects and the projectile objects would be sharing similar properties and in terms of code, methods and variables. The design of the gun and projectile data structure would have a base script containing all the methods and properties that all share between each other as well as unique property methods using generic parameters. This base script will be inherited by the specific scripts for each unique gun and projectile. This would make it easy to make various guns and projectiles with mix and match properties. Concepts for the properties of the guns and projectiles include various firing modes such as burst, single and auto fire and projectile physics including homing bullets, ricochet and bouncing.



Accessibility Features

The accessibility features are designed to be optional measures taken place to allow users to have a more streamlined experience with the software and the medium. There are various methods and concepts designed to address this, the current concepts below are examples of accessibility features I wish to implement in my software.

Snap Teleportation Visual Cover-Up

Covering the user's vision during teleportation can potentially reduce nausea by removing the visual sensory factor in a similar fashion to blinking. The user will not see themselves moving and so will reduce the amount of clashing sensory information thus reducing nausea. This can be with either a dim to black or a flash of white.

Tunnel Vision

Reducing the peripheral view of the user can reduce the sense of motion and in turn reduce potential nausea. This is done by diming and reducing side on the outer field of view of the user's screen, this can be use consistently or on only moments of high speed.

Depth Calculator

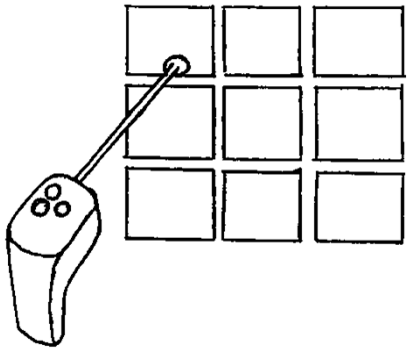
Some users may have difficulty perceiving depth in the virtual reality environment with will cause difficulty in navigating the environment. Including a method for using to gain information on their relative distance to objects in the environment can possibly help the user by giving a defined number to reference to.

UI Design and Interaction

The design of conventional virtual reality controllers is similar to a presentation pointer and UI elements in titles are treated as points on a board to interact with. Buttons, sliders and click 'n' drag objects act the same in the virtual reality otherwise. In-game UI elements that would usually be on a game's Head's Up Display (HUD) have to be treated differently due to the change in viewing angle in a headset compared to a monitor; as the screen is now considerably closer to the user's eyes, objects in the same relative position are prone to go out of focus and thus become illegible to read.

A simple solution to this is change the depth of the HUD in order to make it in focus for the user. This can cause the HUD however to obscure other objects present on the screen, to avoid this the HUD might be shrunk down, but this causes an additional illegibility issue with the HUD being potentially too small. You can also skew the proportions of the HUD to line with the headset's field of view as if the user was wearing a helmet which can address the use of space keeping it out of centre view.

A more dynamic solution is to have UI elements attached to relevant objects, such as ammo count above guns and timers on access by the user's controller wrist. These may feel more natural to take information from as they require conscious action or are present by the object of focus.



Software Implementation

Main development started in conjunction with the fundamental design concepts set up. Changes to design were made during development and the presence of the COVID-19 pandemic cut the available development time for the project. This section will discuss specific systems and methods in the software, the changes made during development and addressing issues and solutions made during development.

SteamVR Integration

Valve developed a programming API for the Unity Engine and the OpenVR framework. SteamVR Integration provides built systems to create a foundation for VR systems in software including a pre-built teleportation system and an object interaction system. A developer can potentially create their entire virtual reality control system with the pre-built assets provided in the API. My use of the API was to add basic functionality as a foundation for my further development. I made use of the SteamVR integration to set up the camera system and controller detection, setting up the backbone for the rest of the virtual reality systems.

SteamVR Action Variables & Input Sources

SteamVR Action Variables and Input Sources are the variables and parameters used by the SteamVR API to directly access inputs made by the controllers. Action Variables are special data types which can be tied to certain buttons and inputs on a controller, these are tied to the Input Sources through the Unity development environment. This can also be used to create custom control schemes which will be useful for development on other headsets or providing users with remappable control schemes.

SteamVR Interactable

The hand game object provided in the SteamVR API is a specialised game object that contains the properties and methods used for interactivity of controllers in the Unity environment. One of the methods determines if any objects are attached to the hands which is primarily used in conjunction with another script known as Interactable. The interactable script provides basic grabbing and object handling options but does not provide features such as force grabbing which was a desired feature to implement – due to development cut times this wasn't able to happen.

Player Movement

The movement system built for the software is like that of a traditional control system. Movement is performed by use of the Rift's left controller analogue stick; rotation is dependent on whether the user has decided to use the Head Movement Dependency or to use the more conventional independent rotation.

```
void PlayerMovementHMD()
    {
            Vector3 direction = Player.instance.hmdTransform.TransformDirection(new
Vector3(playerMovementHMD.axis.x, 0, playerMovementHMD.axis.y));
            characterController.Move(speed * Time.deltaTime *
Vector3.ProjectOnPlane(direction, Vector3.up) - (gravity * Time.deltaTime));
    }
```

For Head Movement Dependency, the method takes the position and rotation value of the headset and uses the z-axis forward value for the Character Controller move method. The independent movement algorithm requires an input source parameter, this is a Boolean accessed through a left or right input on the Rift right controller analogue stick. Detection of a left or right input will be answered with a rotation in the corresponding direction. This is a static rotation requiring multiple inputs; an idea designed post development was to build this algorithm as a continuous rotation, this would be more accurate for the user as they would have access to more precise rotations.

Teleportation Ball System

The teleportation system makes use of the SteamVR hand game objects, interactable system and coroutines. The code first checks whether to see if any objects are currently attached to the hands (play is currently holding an object such as a gun). If the hand is currently empty the player will be able to launch a teleportation ball. Each hand has its own individual version of the method allowing for either hand to be used for teleportation.

```
void TeleportingLeft(SteamVR_Input_Sources source)
    {
        if (leftHand.gameObject.GetComponent<Hand>().AttachedObjects.Count == 0 &&
TeleportBallWorldSpace == false)
        {
            if (teleportTrigger[source].stateDown)
            {
                TeleportBallWorldSpace = true;
                Quaternion projectileRotation;
                projectileRotation = leftTeleSpawnPoint.transform.rotation;

                _ = Instantiate(teleportBall,
leftTeleSpawnPoint.transform.position, projectileRotation);
            }
        }
        else
        {
            Debug.LogError("Object in hand or already thrown ball");
        }
    }
```

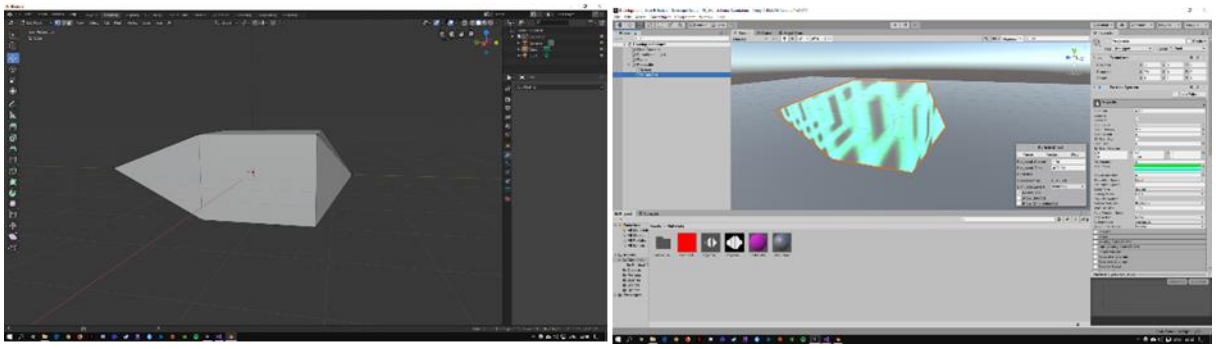
The teleportation ball game object has a small script attached that on collision, identifies what type of surface the object landed on (surfaces are identified by the Unity Engine's tag system). If the ball hits an acceptable surface to teleport on, a method is called in the player script to teleport the player according to their selection in the settings. Due to the cut in development time, only snap teleportation was implemented. For the 'Flash' and 'Blink' teleportation snap systems, each switch case triggers a coroutine method to initiate the visual effect; coroutine is used so the effect can be performed on a set timer.

```
case "Flash":
flashFade.CrossFadeAlpha(1, 0.25f, false);
StartCoroutine(DelayTeleport(0.25f, flashFade));
break;

IEnumerator DelayTeleport(float delay,Image flashType)
{
    yield return new WaitForSeconds(delay);
    characterController.transform.position = TeleportPosition;
    flashType.CrossFadeAlpha(0, 0.25f, false);
}
```

Guns and Projectiles Implementation

The development of the guns and projectile systems as well as programming included the development of the visual assets. The guns and projectiles were developed in Blender and the projectiles where further developed with Unity's particle effects system. The projectiles and guns were designed with a specific colour to correspond with the properties of each gun.



Both systems are built with a generic script from which all gun and projectile scripts will inherit from. The properties are built in which they can be mixed and matched and so are all written into the generic script. These properties are accessed by the specific scripts by get/set variables for which the specific scripts contain the values to write over.

```
//Universal variables for any projectile type
    public float Speed { get; set; }
    public float ProjectileLifeSpan { get; set; }
    public float ReflectionCount { get; set; }
    public float BounceCount { get; set; }
//Toggle variables for various gun properties
    public bool InfiniteLifeSpan { get; set; } = false;
    public bool BouncingBullets { get; set; } = false;
    public bool RichocetBullets { get; set; } = false;
```

The ricochet property is built with a small range raycast in front of the projectile; when the raycast detects a wall collision, the bullet uses the reflect method to rotate the projectile to simulate bouncing off. The bouncing property makes use of the Unity material property system and is added to the projectile on instantiation. Some issues come with the ricochet bullet as if sent at a high enough speed, the bullet can simply pass through objects; an idea that could avoid this issue is to have a pre-determined path system for the bullets. When fired, the bullet could project a raycast with the set distance and reflections that the bullet can follow – these can be also be recalculated every reflection to account for moving objects.

Miscellaneous Development Bugs & Issues

Developing a Visual Aim Reticle

An initial design feature that was made in pre-development was to have a visual aid for aiming the guns within the game. The visual aim reticle was built in two parts, a laser sight and a visual sprite position at the end of the laser sight. The laser sight was simply done with a raycast and a Unity Line Renderer tracing across the raycast, when the raycast collides with a wall the distance is sent to the Line Renderer to change its distance. The main issue came with the sprite which was programmed to update its position to the end of the raycast and rotate its position accordingly. This however wasn't the case as the sprite was stuck in the centre of the development space, it did however rotate according to the position of the laser. Due to the cut development time I was not able to finish fixing the sprite, the solution would probably be however to use a parent game object for the sprite.

Freezing Stopwatch Timer

The stopwatch variable which is built for the course is stored within an overall game script, this script is accessed across all scenes and holds universal information. The stopwatch variable would start by a button on the course and at the end of the course a button to stop the timer. Destroying the targets during the course are designed to freeze the timer for two seconds, this wasn't the case however as the targets would attempt to stop the timer, but the code would be stuck, causing the targets to not be destroyed either. At the time of development this was disabled to work on other priorities but due to development being cut short this wasn't returned to.

Non-Interacting Menu

The menu designed to navigate the settings uses a raycast and line renderer with a Unity event camera system to interact with a canvas in world space. This event camera has to be in front of the Line Renderer in order to interact with the canvas but during initial development, the camera was tied automatically to the headset's event camera which made the headset essentially the pointer. This was a quick fix changing the event camera.

Development Evaluation

The resulting software that was developed is lacking compared to the written requirements I have made; while there are some accessibility options available within the developed software, other designs and concepts I have built did not make into the project. What is present is built quite well, code is arranged quite well though some statements and methods could use some code cleaning. The software developed I feel meets the original project statement however as a game it is considerably hollow.

Conclusion

The development on the project and software if put into a word could be described as problematic. There were multiple factors that slowed down design and pre-development and when full development started, the time available was reduced considerably and the new environment for the short development time didn't provide me with the opportunity to flesh out the software as much as I wanted. I personally have a positive look on the research portion on the project; the research was an opportunity to take a closer look at video game development as well as the virtual reality medium and the information gained from it I feel will be able to be carried over to other endeavours beyond the honours stage project.

Multiple designs and concepts had to be scrapped for this project and while surprise circumstances have been mentioned, a considerable amount of blame can be placed on myself for the results of the project. My work ethic, especially during the beginning of the project, was very low and only increased later and not before long the other factors occurred. All this effected my time management and development methodologies which weren't held – by the end of development, the methodology transformed into a very volatile version of agile programming quickly building parts for demonstration.

If comparing to the described aims and objectives as well as the project task description, I would conclude that I have reach the project task description and some of the objectives. The research objectives I feel have been reached while demonstrating accessibility options within the software is lacking. If provided with more time to continue development, my main goal would probably to implement a new teleporting system, depth calculator and the tunnel vision concept I initially wanted to add; changing the main gameplay level would be a positive option to further demonstrate the additions I have included.

References

Web Articles

- Wawro, A. (2019). *Fine-tuning for the 'action hero' feel in PlayStation's VR game Blood & Truth.* [online] Gamasutra. Available at: https://www.gamasutra.com/view/news/352259/Finetuning_for_the_action_hero_feel_in_PlayStations_VR_game_Blood__Truth.php [Accessed 3 Nov. 2019].
- Murphy, B. (2018). Game Design Deep Dive: Making VR movement feel like a superpower in Gunheart. [online] Gamasutra. Available at: https://www.gamasutra.com/view/news/319596/Game_Design_Deep_Dive_Making_VR_movement_feel_like_a_superpower_in_Gunheart.php [Accessed 4 Nov. 2019].
- Gamasutra. (2019). How Ubisoft designed and refined its Assassin's Creed VR escape rooms. [online] Available at: https://www.gamasutra.com/view/news/352175/How_Ubisoft_designed_and_refined_its_Assassins_Creed_VR_escape_rooms.php [Accessed 8 Nov. 2019].
- Lang, B. (2019). Check if Your PC is VR Ready for Oculus Rift, HTC Vive & Windows "Mixed Reality" VR Headsets. [online] Road to VR. Available at: https://www.roadtovr.com/how-to-tell-pc-virtual-reality-VR-oculus-rift-htc-vive-steam-VR-compatibility-tool/ [Accessed 8 Nov. 2019].
- Kumparak, G. (2014). A Brief History of Oculus. [online] Tech Crunch. Available at: https://techcrunch.com/2014/03/26/a-brief-history-of-oculus/ [Accessed 8 Nov. 2019].
- Hayden, S., 2020. Oculus Reveals More Than 175,000 Rift Development Kits Sold – Road To VR. [online] Road to VR. Available at: https://www.roadtovr.com/oculus-reveals-175000-rift-development-kits-sold/ [Accessed 10 Nov. 2019].
- Edd Gent (2016). Are Virtual Reality Headsets Safe for Kids? [online] livescience.com. Available at: https://www.livescience.com/56346-are-virtual-reality-headsets-safe-for-kids.html [Accessed 20 Dec. 2019].
- Hamilton, I. and Feltham, J., 2020. Revisit the Classic Oculus Tuscany Demo in Boneworks. [online] UploadVR. Available at: https://uploadvr.com/tuscany-boneworks/ [Accessed 8 Jul. 2020].

Research Papers

- Desai, P.R., Desai, P.N., Ajmera, K.D. and Mehta, K., 2014. A review paper on oculus rift-a virtual reality headset. arXiv preprint arXiv:1408.1173.
- Tan, C.T., Leong, T.W., Shen, S., Dubravs, C. and Si, C., 2015, October. Exploring gameplay experiences on the Oculus Rift. In Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play (pp. 253-263).
- Llorach, G., Evans, A. and Blat, J., 2014, November. Simulator sickness and presence using HMDs: comparing use of a game controller and a position estimation system. In Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology (pp. 137-140).
- Davis, S., Nesbitt, K. and Nalivaiko, E., 2015, January. Comparing the onset of cybersickness using the Oculus Rift and two virtual roller coasters. In Proceedings of the 11th Australasian Conference on Interactive Entertainment (IE 2015) (Vol. 27, p. 30).

Games

- BEAT GAMES (2018) *Beat Saber.* [Online Download] Oculus Quest. Beat Games.
- EPIC GAMES (2017) *Robo Recall.* [Online Download] Microsoft Windows. Epic Games.
- DROOL (2016) *Thumper*. [Online Download] Microsoft Windows. Drool.

- GIANT FORM ENTERTAINMENT, LLC (2017) *Aircar.* [Online Download] Microsoft Windows. Giant Form Entertainment, LLC.
- HELLO GAMES (2016) *No Man's Sky.* [Online Download] Microsoft Windows. Hello Games.
- VALVE CORPORATION (2016) *The Lab.* [Online Download] Microsoft Windows. Valve Corporation.
- STEEL CRATE GAMES (2015) *Keep Talking and Nobody Explodes.* [Online Download] Microsoft Windows. Steel Crate Games.

Videos

- CG Cookie - Unity Training (2014). CG Cookie - Unity Training. YouTube. Available at: https://www.youtube.com/watch?v=hUg3UfE186Q&list=PLtOAN1OHEoLYKv_JZGmMeOyblFG-g_LUO&index=5 [Accessed 18 Nov. 2019].
- Dino, D. (2018). How to make Homing Projectiles for your C# Unity Game. YouTube. Available at: https://www.youtube.com/watch?v=t_-kTapjfhM&list=PLtOAN1OHEoLYKv_JZGmMeOyblFG-g_LUO&index=8 [Accessed 18 Nov. 2019].
- Valem (2019). STEAM VR - The Ultimate VR developer guide - PART 1. YouTube. Available at: https://www.youtube.com/watch?v=5C6zr4Q5AlA&list=PLtOAN1OHEoLYKv_JZGmMeOyblFG-g_LUO&index=2 [Accessed 10 Dec. 2019].
- Gabriel Aguiar Prod (2018). Gabriel Aguiar Prod. YouTube. Available at: https://www.youtube.com/watch?v=xenW67bXTgM&list=PLtOAN1OHEoLYKv_JZGmMeOyblFG-g_LUO&index=1 [Accessed 12 Jan. 2020].
- Valem (2020). How to Design for Motion Sickness. YouTube. Available at: https://www.youtube.com/watch?v=DR8QVy2LIj4&list=PLtOAN1OHEoLYKv_JZGmMeOyblFG-g_LUO&index=8&t=0s [Accessed 12 Feb. 2020].