



The Heist VR: An Asymmetric Multiplayer Virtual Reality Game to Improve Immersion

Final Report

DT282

BSc in Computer Science (International)

Philip Toolan

C17433026

Dr. Basel Magableh

School of Computer Science

Technological University, Dublin

06/04/2021

Abstract

Virtual reality has an impressive ability to allow players to escape reality. Recent advances in hardware have given players higher fidelity worlds, increased pixel counts in screens, improved depth and range of sound and even smell. Fields of view have been increased to reduce the “binocular effect”. This project investigates the software side of virtual reality.

There have been some improvements in software such as: better tracking technology and the quality of art assets (to match their equivalent in the real world).

By developing this project, the aim is to use software to improve immersion in virtual reality.

This is accomplished by incorporating asymmetric multiplayer mechanics to invite the real world into the virtual and prevent potential breaks in immersion occurring. The project discovered that this method had a positive effect on immersion.

This finding is limited by a number of factors. The quality of gameplay, the number of participants, lack of in-person testing and the number of games compared.


The project consisted of constructing a game around this multiplayer mechanic and compare it with an existing single player experience. A sample size of eight experienced players were exposed to both games and were then tested by questionnaire after their playthroughs.

The findings are consistent with the hypothesis that asymmetric multiplayer enhances the immersive experience of virtual reality.

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

_____

Philip Toolan

06/04/2021

Acknowledgements

I would like to thank my project supervisor Dr. Basel Magableh, for his constant support, encouragement, calming presence and guidance throughout the project.

I would also like to thank Dr. Bryan Duggan for his generosity in providing equipment and guidance at the initial stage of the project.

I would like to thank the eight players for the sacrifice of playing games in the name of science.

I would also like to thank my family and friends for keeping me sane for this year.

Por último, gracias a Alicia por llorar cuando vio mi proyecto por primera vez y por toda su ayuda y apoyo durante el proyecto.

Table of Contents

List of Abbreviations	8
Table of Figures.....	9
1. Introduction	10
1.1. Project Background.....	10
1.2. Project Description.....	11
1.3. Project Aims and Objectives	11
1.4. Project Scope	11
1.5. Thesis Roadmap	12
<i>Design</i>	12
<i>Development</i>	12
<i>Testing and Evaluation</i>	12
<i>Conclusions and Future Work</i>	13
2. Literature Review	14
2.1. Introduction	14
2.2. Alternative Existing Games	14
<i>Keep Talking and Nobody Explodes</i>	15
<i>Black Hat Cooperative</i>	16
<i>The Playroom VR</i>	17
2.3. Technologies Researched	18
2.4. Other Research	22
Networking Solutions.....	24
Rendering Pipelines	24
2.5. Existing Final Year Projects	25
2.6. Conclusions	26
3. Experiment Design	27
3.1 Introduction	27
3.2. Software Methodology	27

3.3. System Overview.....	28
3.4. Front-End	29
Use Case Diagram	31
3.5. Back-End.....	32
3.6. 3D Modelling Pipeline	32
3.7. Conclusions	33
4. Experiment Development	34
4.1. Introduction	34
4.1.1. Changes from the Initial Prototype.....	34
4.2. 3D Modelling.....	34
4.2.1. Concept creation and Grey-Boxing.....	35
4.2.2. Modelling the Assets.....	36
4.2.2. UV Unwrapping, Texturing and Materials	36
4.2.2. Lighting, Colliders and Scripts	38
4.3. Receiving and Interpreting Input	39
4.4. Player Controls.....	39
4.4.1. Controller Player	39
4.4.2 Virtual Reality Player.....	41
4.4.2.1. Movement and Tracking	42
4.4.2.2. Interactions	43
4.5. Bomb Defusal.....	46
4.6. Infringement Detected Mechanic.....	49
4.6.2. Turret Type.....	49
4.6.2. Seeker Type.....	50
4.7. Conclusions	51
5. Testing and Evaluation.....	52
5.1. Introduction	52
5.2. Testing.....	52

Test Plan.....	53
5.3. Evaluation	54
5.4. Results from the Questionnaire.....	55
5.5. Conclusion.....	58
6. Conclusions and Future Work.....	59
6.1. Introduction	59
6.2. Issues.....	59
6.3. Conclusions	60
6.4. Future Work.....	61
Bibliography	62
Appendix	65

List of Abbreviations

IDC	International Data Corporation
VR	Virtual Reality
GDD	Game Design Document
MRTK	Windows Mixed Reality Toolkit
SDK	Software Development Kit
XR	Virtual and Augmented Reality
URP	Universal Render Pipeline
HDRP	High-Definition Render Pipeline
UI	User Interface
AI	Artificial Intelligence

Table of Figures

FIGURE 1. KEEP TALKING AND NOBODY EXPLODES(11).....	15
FIGURE 2. BLACK HAT COOPERATIVE(13)	16
FIGURE 3. THE PLAYROOM VR(14).....	17
FIGURE 4. PANOPTIC(16).....	18
FIGURE 5. DELL VISOR VR HEADSET(22).....	19
FIGURE 6. OCULUS RIFT HEADSET(23)	20
FIGURE 7. FEATURE DRIVEN DEVELOPMENT(35).....	27
FIGURE 8. TECHNICAL ARCHITECTURE	28
FIGURE 9. FINAL TECHNICAL ARCHITECTURE.....	29
FIGURE 10. START MENU SIMPLE WIREFRAME	30
FIGURE 11. FINAL START MENU WIREFRAME.....	30
FIGURE 12. CONTROLLER PLAYER'S UI WIREFRAME	31
FIGURE 13. USE CASE DIAGRAM.....	31
FIGURE 14. FINAL USE CASE DIAGRAM	32
FIGURE 15. ASSET PRODUCTION PIPELINE.....	35
FIGURE 16. FINISHED WALL MODEL IN BLENDER	36
FIGURE 17. TEXTURE SHEET FOR DESK ASSET.....	37
FIGURE 18. FINISHED WALL ASSET AFTER MATERIALS ADDED IN UNITY	37
FIGURE 19. MODELS IN THE SCENE IN UNITY	38
FIGURE 20. POP-UP FOR CONTROLLER PLAYER.....	40
FIGURE 21. CONTROLLER PLAYER VIEW.....	41
FIGURE 22. XR GRAB INTERACTABLE CODE AS IT APPEARS IN UNITY	43
FIGURE 23. COLLIDERS ON COMPUTER ASSET	44
FIGURE 24. XR SOCKET INTERACTOR CODE AS IT APPEARS IN UNITY.....	45
FIGURE 25. PHYSICS BUTTON CODE AS IT APPEARS IN UNITY	46
FIGURE 26. TIMER CODE AS IT APPEARS IN UNITY	46
FIGURE 27. WIRE CODE AS IT APPEARS IN UNITY	47
FIGURE 28. UNITY HINGE JOINT SETTINGS.....	48
FIGURE 29. FINAL IN-GAME MODEL OF THE BOMB	48
FIGURE 30. TURRET BEHAVIOUR TREE	49
FIGURE 31. SEEKER BEHAVIOUR TREE.....	50

1. Introduction

In this chapter all the background information for the project will be discussed along with the project goals, scope, and roadmap of the paper.

1.1. Project Background

According to the International Data Corporation (IDC), sales of virtual reality (VR) headsets will increase by 81.5% by 2024 (1). Finding new ways to improve virtual reality will be key to keep this growth alive.

One of the compelling factors making VR gaming unique is its ability to immerse the player in a fictional world. This illusion can be destroyed when the player is disturbed by something happening in the real world around them (2). The goal of this project is to make Virtual Reality more immersive by incorporating the real world into the game using asymmetric multiplayer (2).

Asymmetric multiplayer is a type of multiplayer in which the different players have totally different roles and capabilities, unlike most multiplayer games, where all the players are generally completing the same tasks and playing the game the same way (3). In this project, there are two roles in the game which will be referred to as the VR player and the controller player.

The game will require the two players to communicate with each other and to help each other out with tasks in the game to complete their objective. This will make the experience more riveting for the player and reduce the risk of immersion loss due to the real world environment (the controller player) having an impact on the virtual world (4).

There is a number of multiplayer experiences available for VR users like *VRChat* (5), *Second Life* (6) and *Pavlov VR* (7). These experiences do not involve asymmetry in their design however, as they all require the user to be in VR. They go to great lengths to immerse the player in their virtual worlds like in *Second Life* where users can literally live another virtual life where they can buy houses and various other props from fellow users. The platform is so popular that

in 2019, creators on the platform made 60 million US dollars (8). All of this adds to the immersion of the player as they invest completely in their virtual life however as soon as someone in the real world makes a noise, that immersion is broken and can ruin the experience (2). This reason is why a relatively simple VR experience in comparison can be more immersive for the player because it uses asymmetric multiplayer design.

Asymmetric multiplayer games have been done in many unique ways with and without VR. *Evolve* (9) and *SpyParty* (10) would be examples of asymmetric games without VR and one of the most popular in VR is *Keep Talking and Nobody Explodes* (11). We will discuss these and a few more similar games in the research chapter later in the paper.

1.2. Project Description

The Heist VR is an asymmetric multiplayer game that strongly incentivises communication between players in VR and on desktop. Through the use of this communication, it will be able to sustain the immersion of the VR player. The project consists of multiple parts. The VR player, the controller player and the 3D models.

1.3. Project Aims and Objectives

The overall aim of the project is a relatively simple one: to improve the level of immersion in virtual reality by incorporating asymmetric game mechanics and testing their impact on immersion.

1.4. Project Scope

The project consists of a build of the game that demonstrates the goal of increasing the level of immersion. The game has a game design that includes social interaction with a player in the same physical space as a core component of the game loop. The game is built in *Unity*, allowing two players to play on the same machine. The final parts of the project are the assets built in

Blender that are used by the game, the final thesis and a Game Design Document outlining the core game loop and mechanics of the game.

1.5. Thesis Roadmap

Research

This chapter explores background research related to the area of VR multiplayer, immersion and social interaction in VR. Following this, a cross-examination of similar VR games that follow an asymmetric multiplayer approach will be completed. Finally, this chapter will discuss an array of other relevant research completed for this project.

Design

This chapter delves into the methodology chosen for this project and how these choices came to be. Following this, detailed use-cases and wireframes related to the user experience will be presented. Finally, an overview of the approach taken to create the 3D models for the game is discussed.

Development

The development chapter breaks down the entire development process of the system regarding the technical architecture outlined in the design chapter. Some of the challenges encountered in the development process and the development to this point will be explored.

Testing and Evaluation

This chapter describes how all the testing and evaluation of the game was executed. Each phase of testing will be described in detail. How immersion is tested and evaluated will be discussed and the results of testing will be explored.

Conclusions and Future Work

This chapter will reflect on the entirety of the project and will discuss the conclusions drawn, personal reflections made, and the future work recommended for the project.

2. Literature Review

2.1. Introduction

In this chapter some of the key areas of research that are important in this project will be presented. These topics include a cross-examination of similar VR games that follow an asymmetric multiplayer approach. Following this, the relevant technologies researched will be explored. Finally, this chapter will discuss an array of other relevant research completed for this project.

2.2. Alternative Existing Games

The idea of asymmetrical multiplayer games has been around for years, but because VR is still in an early stage (12) there is not many games in this style on the platform. Some of the games used as inspiration are *Keep Talking and Nobody Explodes* (11), *Black Hat Cooperative* (13), *The Playroom VR* (14) (15) and *Panoptic* (16). These games take the same approach to asymmetric multiplayer by having the VR player experience a different side or part of the game to the controller players.

The research was mainly focused on these asymmetrical games as they are the closest pieces of existing software to the project. Research was also done into other VR games however, to find other sources of inspiration for the project and ensure a wide range of research was conducted. These games include *Second Life* (multiplayer), *VRChat* (multiplayer), *Half-Life: Alyx* (17) (single player) and *Blood & Truth* (18) (single player). Looking into these games ensured that the design approach for VR and multiplayer in VR would follow a similar standard to the VR gaming industry. The research findings from these games will be included in the “Other Research” section of this chapter.

Keep Talking and Nobody Explodes

In *Keep Talking and Nobody Explodes* the player in VR examines a bomb and tries to diffuse it with the help of the controller players. The controller players never see the bomb, but they have an instruction booklet on how to diffuse it and must help the VR player diffuse it within a time limit. This design incorporates the outside world in the game by forcing that communication and making the VR player rely on the controller players and vice versa. One problem with this however is that the VR player is the only player able to interact with the fictional world, this can make the player still feel isolated (19).



Figure 1. *Keep Talking and Nobody Explodes*(11)

Black Hat Cooperative

Black Hat Cooperative is the game with a design that is most similar to this project. The VR player infiltrates a facility while the controller player helps them achieve this. Both players are able to interact with the virtual world and they can communicate to complete the objective. The controller player can see enemies and can unlock doors for the VR player who sneaks through the level. The communication part of the game is what makes it the most fun (20) but it is not a requirement at many stages throughout the game. It is important that the communication between both players is intrinsic to the game experience in order to keep the level of immersion high (21). For this project it is therefore important that it provides a lot of opportunities for communication between players.

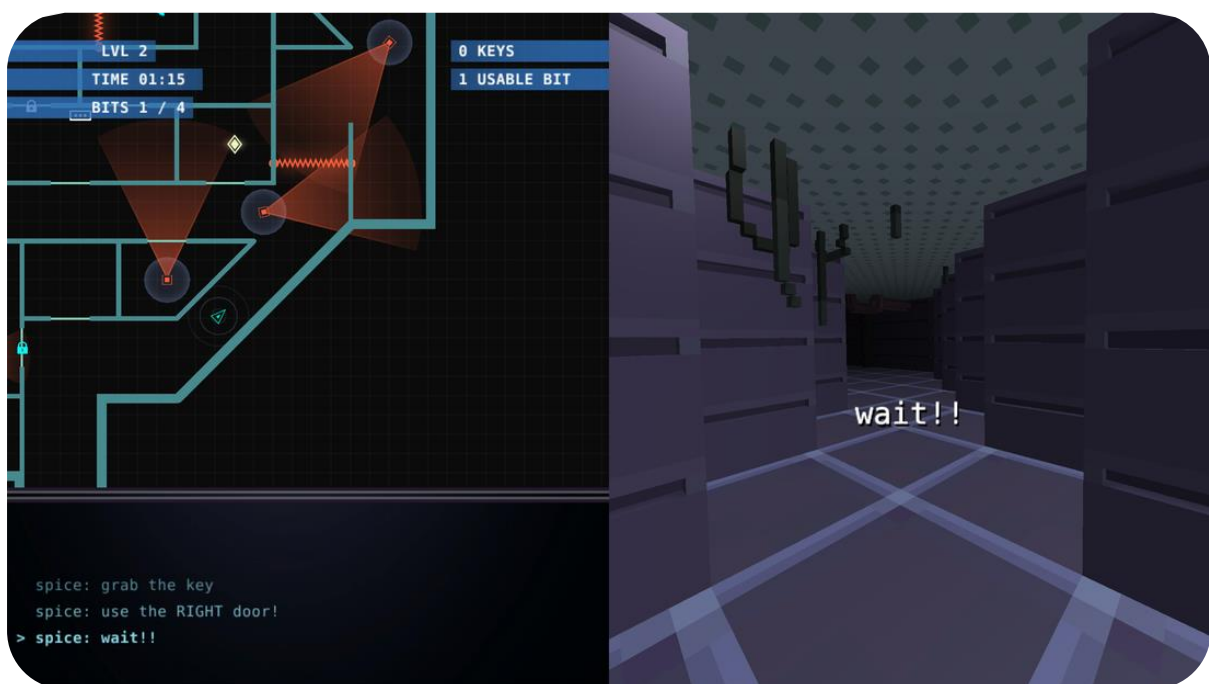


Figure 2. *Black Hat Cooperative*(13)

The Playroom VR

Playroom VR works well at giving the VR player a different perspective to the controller players. The game allows both sets of players to interact with the fictional world in meaningful ways, but they do not set up the need for communication. They set up a good social environment for the players, but they fail to capitalise on this opportunity by making it competitive instead of cooperative or by making the cooperative scenarios too simple to require communication.



Figure 3. The Playroom VR(14)

Panoptic

Panoptic takes an entirely competitive approach to asymmetrical multiplayer in VR. The VR player is tasked with finding the controller player amongst dozens of non-player characters that look the same. The controller player must interact with several different beacons which will reveal their position to the VR player, before leaving the level. This method does not encourage discussion between the players but does explore asymmetric game design in VR.



Figure 4. *Panoptic*(16)

2.3. Technologies Researched

Game Engines

For this project, two of the most popular game engines in the industry for VR titles were investigated: *Unity* and *Unreal*.

Unity is a free to use game engine that uses the C# programming language. *Unity* has a massive depth of tutorials and resources available for free on their website which can help greatly with the project. It has several plug-ins that can be used to ease the development process for VR, and it is popular among small development teams in the gaming industry.

Unreal is another game engine that was researched for the project. The engine uses the C++ programming language and like *Unity*, they offer a wide range of tutorials and resources online. *Unreal* offers similar plug-in capabilities to *Unity* with regards to VR. The engine is very popular across the industry however it is seldom used by independent developers or small development teams. Some of the reasons for this is due to the hard to learn user interface and the ease of use of the *Unity* game engine. This is why the *Unity* engine was chosen for this project.

Virtual Reality Headsets

For this game, the *Oculus Rift* and *Dell Visor VR* were researched for the project.

The *Dell Visor VR* is a headset developed by Dell for the Windows Mixed Reality platform. The headset features “inside out tracking”, this means that the headset can track your position in game using two cameras on the front of the headset. This makes the headset significantly easier to set up and develop for, however, it is not as reliable as more traditional tracking methods that will pick up all your movements. The headset is compatible with the SteamVR platform and can be easily developed for in *Unity*.



Figure 5. Dell Visor VR headset(22)

The *Oculus Rift* headset is one of the leading headsets in the VR industry due to its low price and high quality. The *Rift* has been usurped by the *Rift S* and *Quest* from Oculus since its inception in 2012. The *Rift* is outdated by today’s standards and it requires “base stations” that

are needed to track the position of the VR player at all times. Also, due to the Coronavirus pandemic it was difficult to retrieve a headset from the college. The easier use of the Dell Visor's "inside out tracking" and the lack of availability for the *Oculus Rift* are why the *Dell Visor VR* was ultimately chosen for the project.



Figure 6. Oculus Rift headset(23)

3D Modelling

For the project, three different modelling software were researched. These three were chosen due to their compatibility with *Unity* and their popularity in the gaming industry. The three software are *Blender* (24), *3Ds Max* (25) and *Maya* (26).

Blender is a free and open-source 3D computer graphics software that has numerous uses from visual effects to animation. Despite being open source, the software has full time developers continuing to support it with the help of investors such as *Unity*. There is lots of resources

online to learn the software and the pipeline from *Blender* to *Unity* is relatively simple (27). *Blender* is the software chosen to develop the 3D assets in the project due to the massive online support for the software and its simple pipeline.

3Ds Max is a software solely focused on building 3D models. It can be quite difficult to learn from scratch similar to all other 3D modelling software however it requires a subscription to use. The pay wall for the software means there is less resources online to learn about the software and the process of importing the assets to *Unity* can be difficult to do.

Maya is software that is mainly focused on animation, but 3D models can also be built in it. *Maya* recently became a subscription-based software. The user interface is similarly complicated to the other software researched but unlike *3Ds Max* there is a lot of resources online to learn *Maya*. The high cost of *Maya* makes *Blender* an obvious choice for the project and it was ultimately decided to stick with *Blender* for the project.

Software Development Kits

Windows Mixed Reality Toolkit (MRTK) is the software development kit (SDK) developed by Microsoft for Windows mixed reality headsets like the *Dell Visor VR* chosen for this project. It offers a wide range of pre-built tools for interactions, hand tracking and user interfaces. The documentation of the software is extensive, and it is kept up to date entirely by Microsoft.

Due to the fact the SDK is developed by Microsoft, it focuses on Microsoft products like the HoloLens which is a mixed reality device rather than a VR only device. Upon testing the SDK, it was discovered that many of the tools developed in this SDK are focused completely on mixed reality and is very limited for VR only applications. In addition to this, because *Unity* is not involved in the development of this SDK, many of the tools can become depreciated in newer versions of *Unity*. For these reasons, this SDK was not chosen for the project.

SteamVR was the next SDK explored for the project. It is a feature rich SDK that is developed by Valve. *Unity* used to provide support for this SDK but stopped this support in 2020. Since then, Valve has been the sole developer of the SDK and similar to MRTK, many of the tools developed in the SDK can become depreciated or entirely change in functionality. Despite this,

the SDK does offer more tools for VR than the other SDKs researched. These tools can allow for a quick development of a demo however they restrict the room for features developed from scratch as new features must use these tools even if they contain functionality not needed.

Due to the lack of support from *Unity* and the lack of freedom in developing custom features SteamVR was not chosen for the project.

Unity XR Interaction Toolkit is the SDK developed by *Unity* for VR and augmented reality applications. There is extensive documentation provided by *Unity* for this SDK and there are many tutorials online to learn its tools. Out of the SDKs researched for this project, this had the least number of tools available however they were basic enough that building on them was easiest.

2.4. Other Research

Multiplayer in VR

Multiplayer VR games are rather common in the industry. From life simulations in *Second Life* to shooter games like *Pavlov VR*.

The focus of games like *Second Life* and *VRChat* is player expression. They give the player as much opportunity to express themselves with avatars, clothing and giving them the chance to converse with hundreds of other players across different scenarios. The common goal of all the games is to immerse the player in a world and let them escape the real world. Be that with giving the players lots to do and letting them express themselves or having tight realistic gameplay that makes the player's presence in the virtual world feel real. The only difference between these games and their single player counter parts is that you can play with friends, they lack a real opportunity to improve immersion with asymmetric game design.

Game Design in VR

Game design in VR can be quite different to when designing for ‘flat screen gaming’. The virtual world has to have real life scale, or it would feel completely off to the player (28). Very simple tasks in games like picking up a cup and throwing it become incredible in VR as the player feels the world as being more tangible (29). It is important therefore that in this project the design tries to be as true to life as possible.

In *Half Life: Alyx* the opening of the game is slow compared to traditional games to allow for the user to get acclimated to being in VR. The section has lots of objects to throw and break and even draw with markers. The player can spend all day in this section playing with the everyday objects and getting used to VR or if they are a seasoned VR player, they can walk through it in one minute (29).

While designing *Blood & Truth*, the development team found that they needed to utilise smart placement of enemies and props due to the limitations of VR (28). Enemies needed to always be in front of the player so they would not get tangled in the wires or lose tracking of the player. Door placement in the levels also had to be carefully managed. They had to ensure that they always faced the player so again they would not lose tracking or have the player be confused with the movement system in the game (28).

Locomotion in VR

Locomotion in VR has been done in numerous different ways in the industry. The two most common methods of movement are continuous and teleportation (30). Continuous movement is like any other game, the player pushes the joystick on the controller, and they move. The problem with this kind of locomotion is that the player can feel motion sick very easily (30). Teleportation is the desired method of movement for most players as it allows them to move through the world without feeling sick. They simply point at a point in the game and then press to move there.

Teleportation is going to be the default method of locomotion used in the final version of the project in order to prevent motion sickness.

Networking Solutions

In the initial stages of the project, it was planned to use a networking solution like *Photon*(31) or *Mirror*(32). These solutions are quite powerful and would make the networking aspect of the project considerably easier. The main issue with these platforms is their lack of flexibility.

It was decided to create a custom network solution so that it will be possible to achieve all the features of the project. On top of this, having a custom network solution allows the project to use localhost. This would mean that when playing, the players would not need an internet connection and latency would be reduced.

During the development phase of the project the network was taken out as the server was unable to handle inputs from the controller and VR players simultaneously.

Rendering Pipelines

The *Unity* game engine has several different rendering pipelines for projects available. These include the default render pipeline, the Universal Render Pipeline (URP) and the High-Definition Render Pipeline (HDRP) that were all looked at for the project. The default render pipeline was used at the start of the project due to its initial ease of use however the visuals of this render pipeline are very dated in today's landscape of VR games, the lighting is poor, and performance can be lacking when there are many game objects in a scene. URP is the render pipeline chosen for the project; it improves the visuals greatly thanks to its improved real time lighting. A huge performance boost can be seen compared to the default render pipeline when there are many objects in the game. The final render pipeline look at was HDRP, it has the best visual capabilities of all the pipelines however the difference between HDRP and URP is negligible with the art style of the game. It was decided to stick with URP as the visual improvement with HDRP was not worth the trade off in performance.

2.5. Existing Final Year Projects

Final Year Projects from previous years were consulted in the research phase of the project. I focused on projects with some relevance to the area of VR and *Unity* development.

Enhancing Immersion in Virtual Reality – Peter Flanagan

This project took a different route to enhancing immersion by building a final game that used external hardware peripherals to achieve greater immersion by allowing the player to play in an intuitive way (33). The game does try to bring the world into the virtual world by using these real tangible objects, but immersion loss can still occur from sounds and actions in the real world.

The *Unity* game engine was used to develop the game and a custom-built haptic gun controller was built for the player. The game was able to provide feedback to the player through the gun at the correct times. This effect tries to bring the virtual into the real world rather than the other way around.

Results from the project were positive but the game design left much to be desired as players felt confused about the movement in the game.

Dragon Simulator – Keith Nevin

The goal of this project was to develop a simulation of the experience of riding a dragon in VR. The project strayed from this initial intention by focusing on procedural generation and the scope ended up being too big. Despite this, the results turned out well for the project (34). The dragon in the game was rigged and animated in *Blender* and the experience was built in the *Unity* engine. The lack of a Game Design Document (GDD) hurt the project as the experience lacked depth and was not fleshed out.

This project must learn from this and a GDD will be created so a core game loop can be implemented and increase the depth of the project.

2.6. Conclusions

With the necessary gained knowledge of VR multiplayer game design, Immersion and social interaction in VR and inspirations for game design, the development stages of the game could begin.

After looking at the research and dissecting the good and bad features of inspiration, it was possible to create an appropriate scope for the project.

The technologies best suited for the project were decided after researching all the different options discussed and comparing the complexity of each option.

3. Experiment Design

3.1 Introduction

Following on from the previous chapter, where some of the key background research was presented, these themes will be continued in this chapter, where the design of the game will be presented. The first section will look at the software methodology employed in this project which describes what methodology was chosen and why. After that, an example use-case will be presented. The next section outlines the technical architecture of the system and will discuss in depth how the system architecture works. This will cover both front-end and back-end aspects of the system. Finally, the last section will explain the process to developing 3D models in *Blender* and how these will be ported to *Unity*.

3.2. Software Methodology

A **feature driven development** approach was used where a feature was designed, developed, and tested. Once that feature was implemented, development of the next feature could begin.

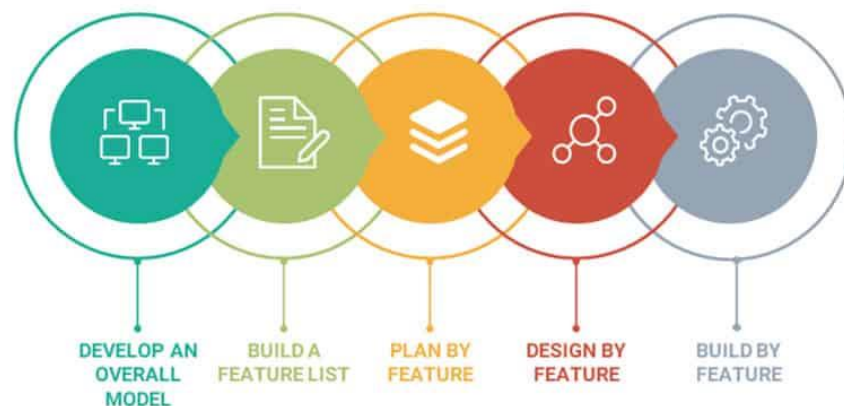


Figure 7. Feature Driven Development(35)

This approach was necessary for this project as features are the main driving point.

Design and code were delivered in stages. A feature was thoroughly designed and researched before implementation began. It would not suit this project to complete all the design up-front

followed by all coding as the codebase is quite large, complex and hard to manage if done all at once.

The general approach for the project was as follows:

1. Design a feature (for example a new game mechanic)
2. Implement the feature.
3. Test the feature.
4. Create and implement desired assets for the feature.
5. Final test of the feature combined with other developed features.
6. Repeat steps 2-5 implementing all desired features – implemented by priority of the features.

3.3. System Overview

The technical architecture shows how many layers there are in the application and how the layers communicate with each other. For this project, a 2-Tier model was originally used as seen below. This model was the original choice as changes to one layer of the model should not affect the other layers.

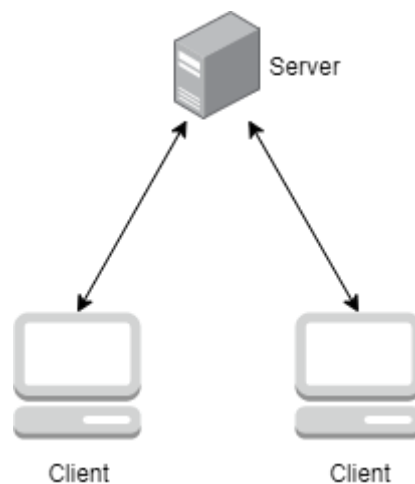


Figure 8. Technical Architecture

Each client was an instance of the game, one in VR and the other on desktop. The server was just a console app running on the computer that facilitates communication between each instance of the game. Certain operations were done locally on the client and others were done on the server. For example, the VR player's camera was done entirely locally on the VR player's client as the controller player does not need to see this.

This original model was altered during the development stage of the project. It was discovered that the server was not able to handle the two control schemes (VR and Desktop) at once. The current architecture consists of one instance of the game running where both players are in the same instance but see and control different aspects of the environment. This can be seen in the model below:

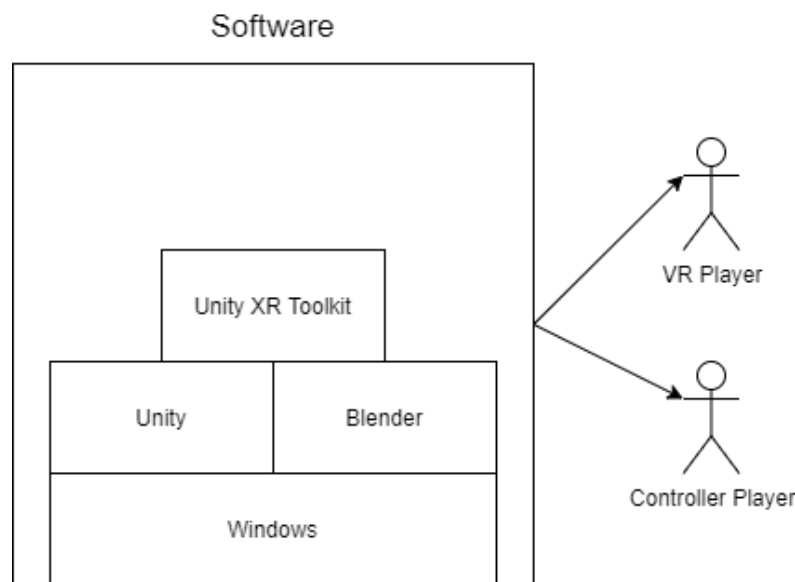


Figure 9. Final Technical Architecture

3.4. Front-End

The front-end layer in the system consists of the user interface. This layer will allow the user to interact with the game world and navigate through the menu of the game. It is important to note that two different interfaces were needed to be made for the VR player and the controller player.

Simple wireframes were used for the first iteration of the menu screens and to provide a low fidelity prototype. These screen mock-ups were useful as they were shown to potential users and gave them a feel for the desired product.

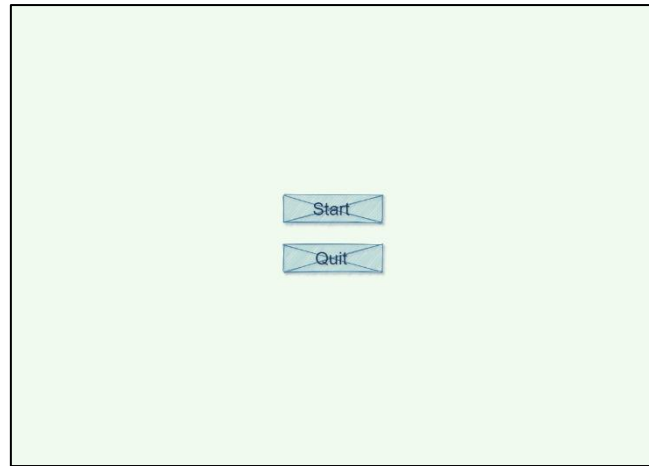


Figure 10. Start Menu Simple Wireframe

This also enabled necessary feedback at an early stage in the design process. Some feedback was received from early testers. Extra features were proposed. These features included adding a bit more support for VR users as some of the options were deemed too small for a VR user to read or select. The final design of the UI took inspiration from other VR games that were researched for the project. The controller player's UI was based on the interfaces from these games and some other non-VR strategy games that require some of the same interactions from the player.

It was decided to leave the main menu exclusive to the VR player, this allows the player to get used to navigating a VR environment and VR controls in their own time.



Figure 11. Final Start Menu Wireframe

The controller player's UI was based around input from the mouse to keep it as simple and intuitive as possible. The player would be able to learn the interface and mechanics of their part of the game while the VR player is interacting with the main menu. The controller player receives information from pop-ups that can then be relayed to the VR player.

1

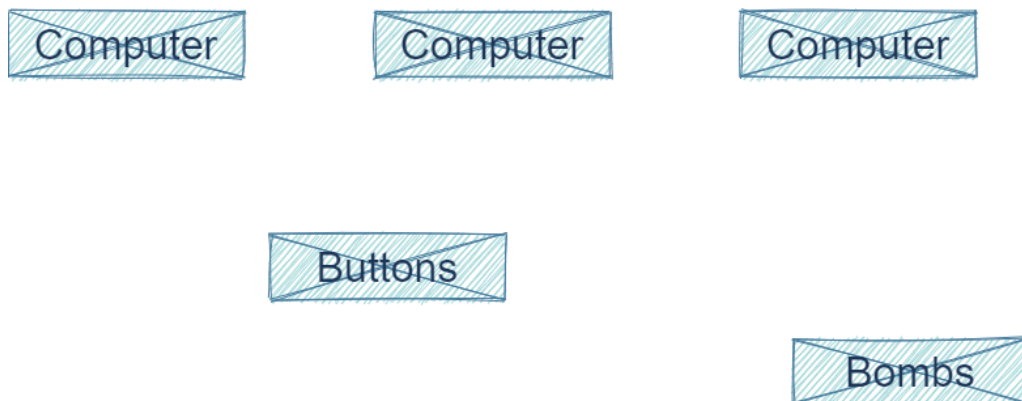


Figure 12. Controller Player's UI Wireframe

Use Case Diagram

Use case diagrams are used to identify system functionality and communicate system behaviour. The use case diagram in figure 13 below shows the original planned system functionality.

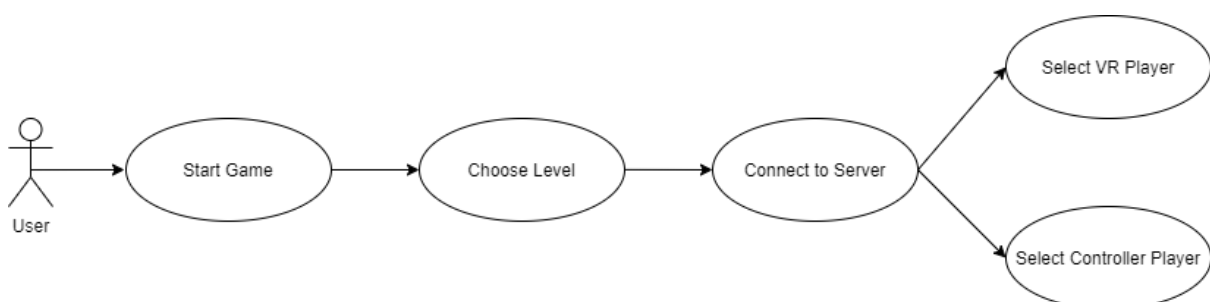


Figure 13. Use Case Diagram

The below use case diagram shows the latest design for the project which has been altered significantly from the original:

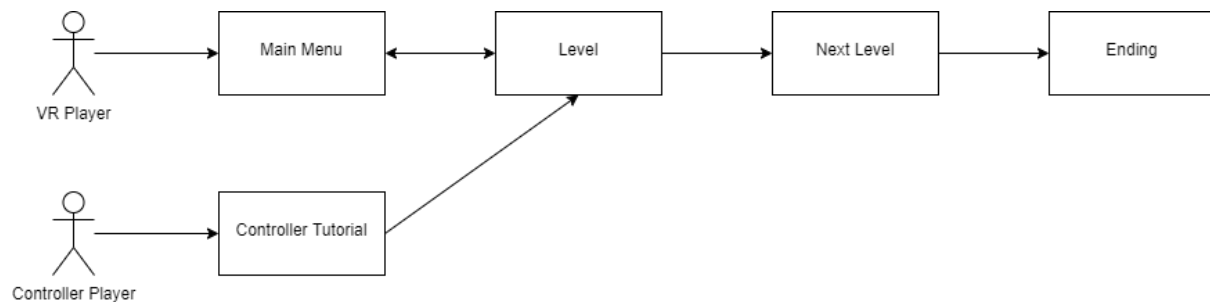


Figure 14. Final Use Case Diagram

3.5. Back-End

The back-end provides the underlying functionality for users to select which level to pick, interact with the environment and gather information. This layer also handles all the game logic and artificial intelligence (AI).

Unity is used to handle all the scripts that control the user movement, AI and game logic. The *Unity XR* interaction toolkit is used to handle the tracking of the VR components like hand tracking and head tracking.

The new *Unity* action-based input system is used so that users with other headsets like the Valve Index or *Oculus Rift* would be able to test and play the game. This design choice was essential as it allows for much more testing during the current pandemic as the project could be sent to users regardless of their VR headset.

3.6. 3D Modelling Pipeline

3D modelling is the process of creating a visual and digital representation of an object (27). A box modelling technique was used to create the models for this project. This technique consists of building the models by using simple shapes. It was important to make sure the models have

a low number of polygons so that when imported into *Unity* they would not be very taxing on the system. Ninety frames per second is the minimum requirement for the *Dell Visor VR*.

The next step was UV mapping. This is the process of projecting a 2D image to a 3D model's surface for texturing (27). This was done within *Blender* as it has UV mapping functionality built in, however it was only required for a small amount of assets in the project.

The next part of the process is rigging. This is where bones are added to the model to make parts of it flexible and moveable so animation can be done. Most of the models for this project do not need this part of the process. For models that needed to be animated, this was done within *Blender* and were saved with the model when it was imported to *Unity*.

The final part before importing into *Unity* is the texturing. This part just adds more detail to the model using the UV map produced earlier in the process. Some materials were also prepared for the models at this stage because *Blender* allows for some finer tuning with materials than *Unity* allows.

Importing into *Unity* was straightforward and the assets were saved in a folder within the *Unity* Project. Some of the materials were further edited in *Unity* to allow for some of the special functions of the Universal Render Pipeline in *Unity*. For instance, emission needed to be added separately in *Unity* for the light on the computer model, this allowed the light to glow in the game.

3.7. Conclusions

The design of the system was looked at in this chapter, first exploring the methodology that was used in the development process, next a broad overview of the technical architecture was outlined, then the details of the front and back-end design. Finally, the approach to the process of creating the 3D models was discussed.

Based on the key themes discussed in this chapter, the next chapter will cover the development process and will be revisiting many of the same issues covered here. The development chapter will discuss how these designs were implemented including any challenges or changes encountered along the way.

4. Experiment Development

4.1. Introduction

This chapter continues with the issues explored in the previous chapter and will outline the development process undertaken in this project. This chapter will present the key development processes and the challenges encountered during the creation of this system. Here is link to the GitHub Repository set up for the project: <https://github.com/PhilToolan/Final-Year-Project>

4.1.1. Changes from the Initial Prototype

After the initial demo of the prototype further development took place. The initial prototype had no VR functionality, implementing this proved to be extremely difficult. Custom networking code was written to allow two instances of the game to communicate, it was originally planned that one of these instances would be VR and the other be the controller player. Porting the VR SDK to the server was going to be impossible to do within the time of the project, issues of detecting VR hardware continued to cause problems while no visible errors could be seen to try and debug the problems.

For this reason, the architecture of the project was altered in late January 2021 so the project could be completed. The current implementation of the project takes place in one instance of the game, the controller player using the desktop and the VR player on the headset. The core goals of the project remained unchanged despite this change in development and design.

4.2. 3D Modelling

All of the 3D models were either made in *Blender* or used simple shapes from *Unity*. For the style of the assets, it was important to keep it to a low poly style. This style is common among many popular VR games like *Job Simulator* (36) or *Budget Cuts* (37). It is popular thanks to its low cost on performance and the ease at which it is to create these models. They do not

require incredibly detailed textures and complex geometry, so it was possible to make these kinds of assets in the time available for the project.

This art style also escapes the trap of the “Uncanny Valley” where assets can be incredibly realistic looking but are just that little bit off and end up being distracting and ruining immersion. The assets focus on the important details of objects and our minds can fill in the gaps, so the assets feel more realistic despite not having painstaking detail.

The style relies on basic geometric shapes so building the assets can be made rapidly on a basic level before more detail is added.

The asset production pipeline shown in the below figure began with creating the concepts from looking at inspirations and grey-boxing the environment. The models were then created in *Blender* and then some of them were UV unwrapped and textured while others only required materials to be made for them. These were then placed in *Unity* where lighting, post-processing, colliders and scripts could be attached to them. These assets would then replace the simple placeholder cubes that had been used to develop the scripts and functionality of the assets.

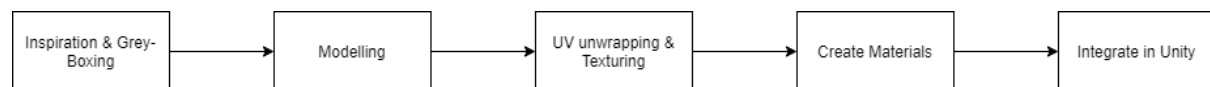


Figure 15. Asset Production Pipeline

4.2.1. Concept creation and Grey-Boxing

Several reference images were gathered during the research, design and development stages. These were taken from a number of VR games and non-VR games that were used as inspiration for the game. These reference images were then used to create a grey-box environment in *Unity*. Grey-boxing is a technique where you can get a general sense of the environment and assets before refining. Once the initial grey-box environment was made in *Unity*, some level design could be done, and it was possible to figure out what assets were required for the environment and the game as a whole.

Through this process, it was determined what assets needed to be made in *Blender* and what assets could be created using *Unity*. Colour palettes from the reference games were also experimented with at this stage to see what colours would suit the environment. These were tested by applying temporary materials to the assets that could be used across the level.

This process went through several iterations over the course of the project in order to test different level layouts, various props, and architectural structures.

4.2.2. Modelling the Assets

Upon discovering what assets would need to be developed and what colours should be used, it was time to move to *Blender* where the assets could be made in more detail using a box modelling technique where the assets are all formed from simple shapes. These models were then saved in *Unity* rather than explicitly exporting and importing the assets. This method allowed for further changes to be made to the models in *Blender* and then almost instantly see these changes in *Unity*.

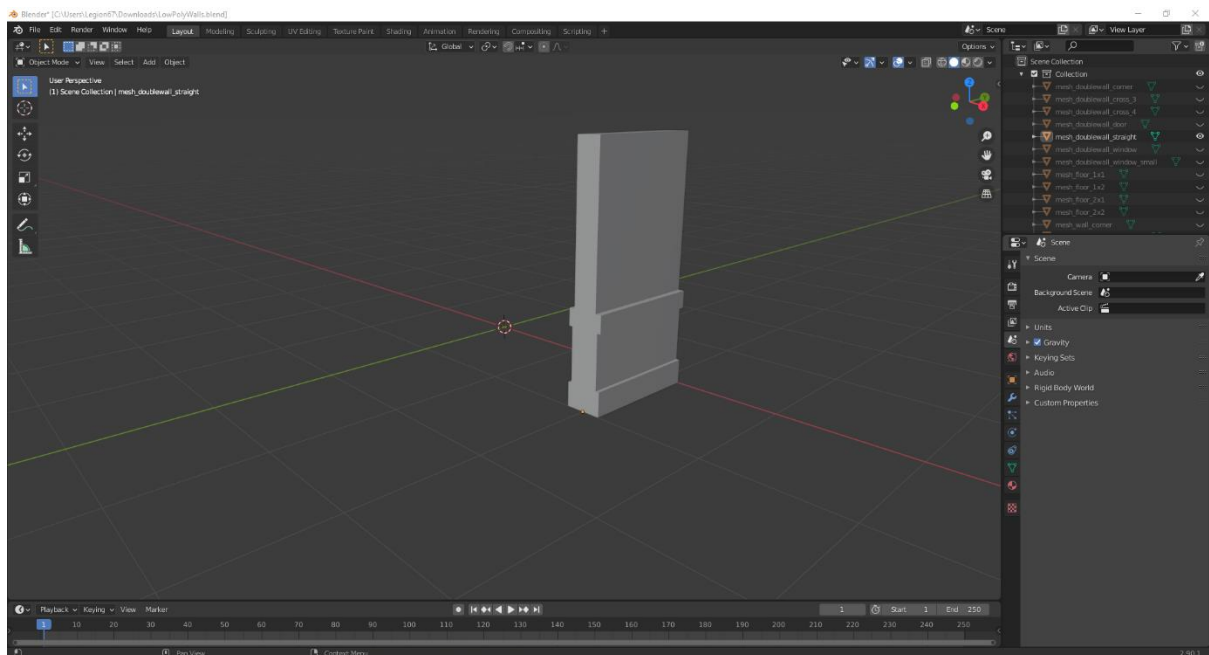


Figure 16. Finished Wall Model in Blender

4.2.2. UV Unwrapping, Texturing and Materials

For the assets that needed to be textured like the desk asset, the UVs were unwrapped for the models in *Blender*. These can then be repositioned to best utilise the texture space and they are saved on a 2D plane. This was then imported into Photoshop where in the below figure you can see a texture applied to certain elements of the model. This process was reserved for assets

that would require attention from the user as they would stand out compared with other assets in the scene.



Figure 17. Texture Sheet for Desk Asset

Materials were first tested out in *Blender* where the colours could be looked at in detail, but all the final materials were made in *Unity*. The materials followed the test designs in *Blender*, but further details were added like emission and how light would reflect off the objects.

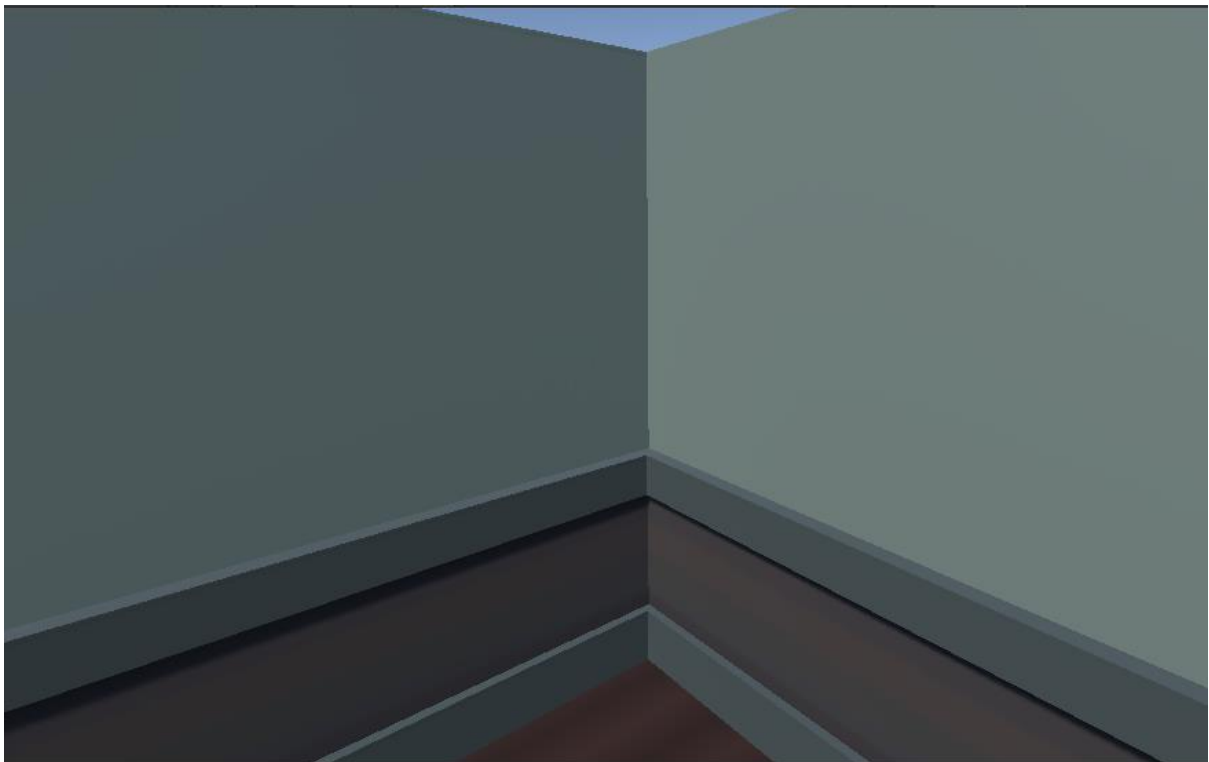


Figure 18. Finished Wall Asset After Materials Added in Unity

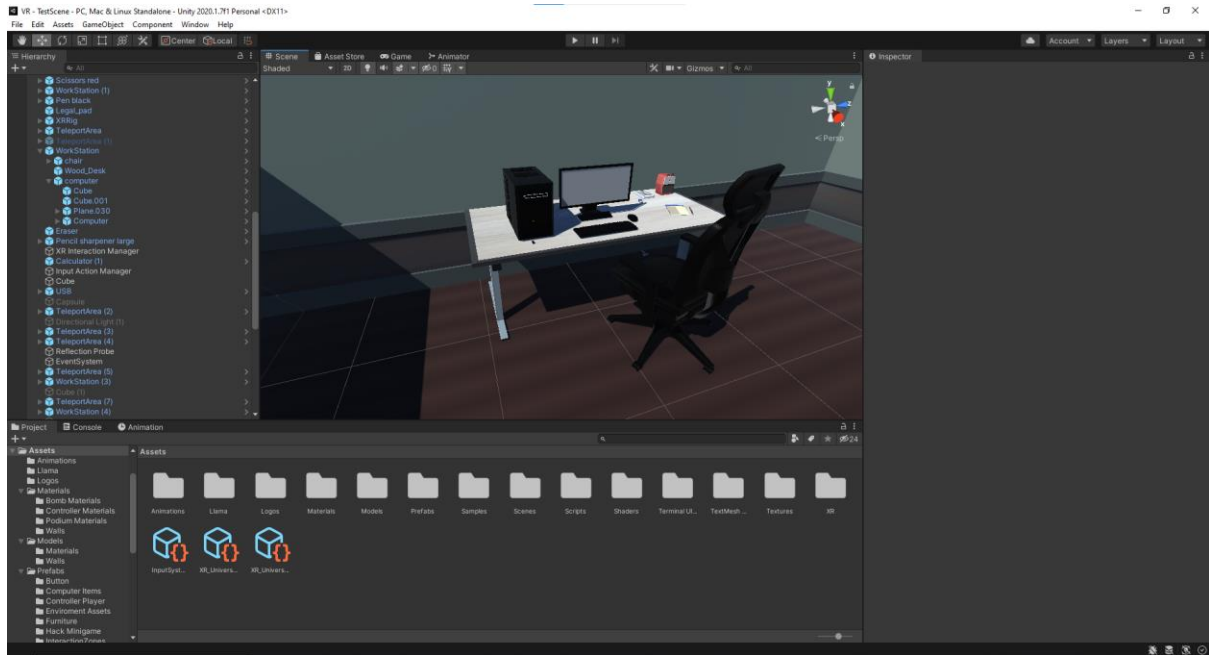


Figure 19. Models in the Scene in Unity

4.2.2. Lighting, Colliders and Scripts

The final steps for integrating the assets and creating the environment in *Unity* were adding lighting to the environment, colliders, and scripts to the models. As the game requires 90 frames per second to work properly, the lighting had to be particularly optimised. The game sticks with baked lighting rather than real-time lighting which can be seen in most games. In real time lighting, the effect of each light on the objects around it are calculated every single frame before it is displayed to the screen. This can have a drastic effect on performance when you start to add all the lights in the scene. When using baked lighting like in this project, *Unity* generates a lightmap in the editor before runtime. *Unity* calculates all the lighting for the entire scene and determines what should happen to each asset, like brighten them or add shadows. These calculations are added to the lightmap this lightmap is merged with the original textures and materials at runtime. This vastly improves the performance of the game with minimal cost as dynamic lighting is not necessary. This prevents lights from being moved in the scene, however this never became a problem so baked lighting was used throughout.

At this stage, colliders and scripts were also added to each asset. The colliders are what prevent assets from falling through each other and scripts provide the functionality of the assets. These will be discussed in further detail later in the chapter.

4.3. Receiving and Interpreting Input

Input is received and interpreted using two different methods. The VR player uses *Unity*'s new action-based input system while the controller player's input is handled by the old *Unity* input system. The reason why two different input systems were used is because they are both good at certain things.

The action-based input system is good at interpreting input across platforms and devices, it is also the supported input system for the latest version of the *Unity* XR Interaction Toolkit. It works across platforms easily as you set up one 'action' and the buttons for this action can be set across any and all devices. Therefore, the layout of different VR controllers does not matter, and the game would still be intuitive to control. As this input system is also integrated into the *Unity* XR Interaction toolkit, setting up a scene can be quick, and interactions can be implemented easier than if the old device-based interaction system was used for VR.

The old input system is used for the controller player as it is quicker to set up, will not confuse inputs with the VR player and the controls are so minimal that it is unnecessary to adapt them to other platforms (e.g., Qwerty keyboard vs Qwertz keyboard) as the differences do not affect the controls. The controls accept mouse clicks and 'horizontal and vertical' input. *Unity* has a built in horizontal and vertical input that it can accept, it accepts arrow clicks or on a qwerty keyboard it would also accept WASD as the inputs on the horizontal and vertical axis. As the controller player's inputs are centred entirely around these, they are unable to accidentally affect game objects in the VR player's world and vice versa. This allows the single instance of the game to be possible.

4.4. Player Controls

4.4.1. Controller Player

The main challenge of developing the controls for the controller player was making it intuitive to control, designing an effective user interface (UI) and giving good feedback to the player.

The target audience of the game is very varied so keeping the controls for the controller player simple was a key aspect to keep in mind. Unlike the VR controls, controlling a game on a desktop can be quite daunting for newer players. The controls for the controller player therefore are centred around the use of a mouse and minimal keyboard inputs.

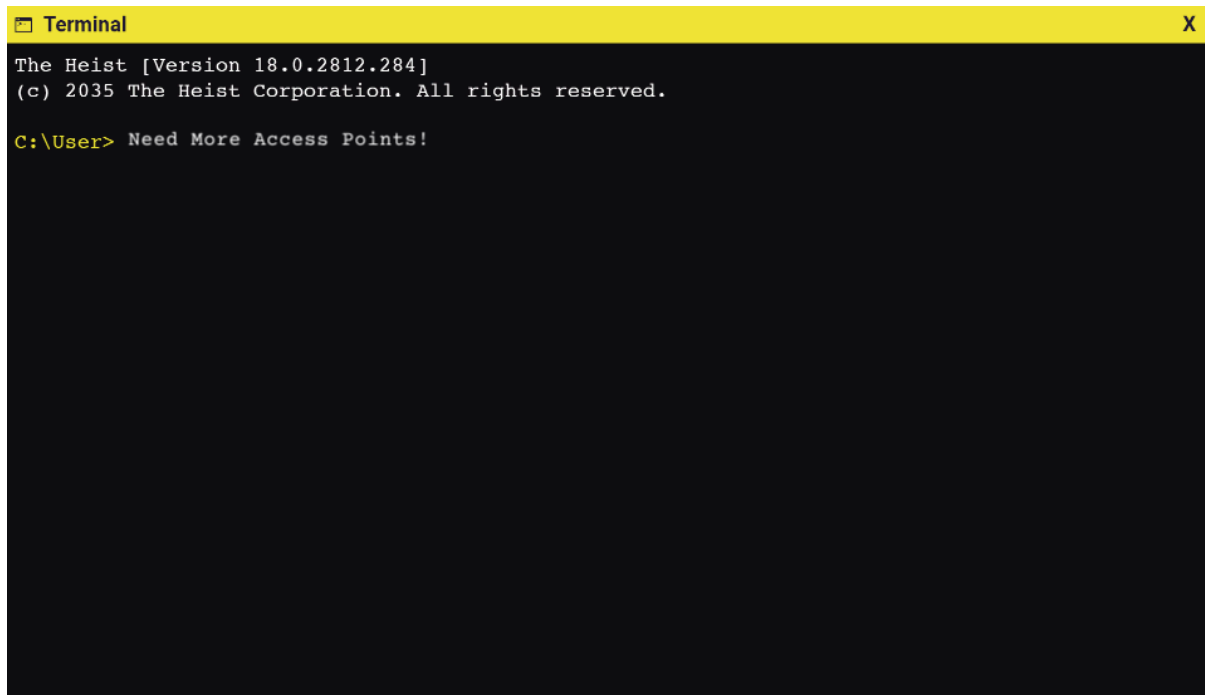


Figure 20. Pop-up for Controller Player

The skills of the controller player are focused on navigating the menu and communicating with the VR player effectively, their skills in gameplay are only tested in small “hacking minigames”. This lack of focus on gameplay for the controller player allows them to communicate easier and further enhances high stress situations where gameplay is introduced with communication to make it harder for the player.

The controller player sees a stylised isometric view of the VR players world to symbolise them hacking into the building. Interactable items are in red, the controller player can click on these with their mouse to receive information about the item. This information can then be communicated to the VR player as instructions to continue to progress in the game. Yellow items are game objects with which the Controller player is currently unable to interact. To gain access to these items, the controller player would have to direct the VR player to an ‘access point’ that would give the controller player the ability to interact with the object.

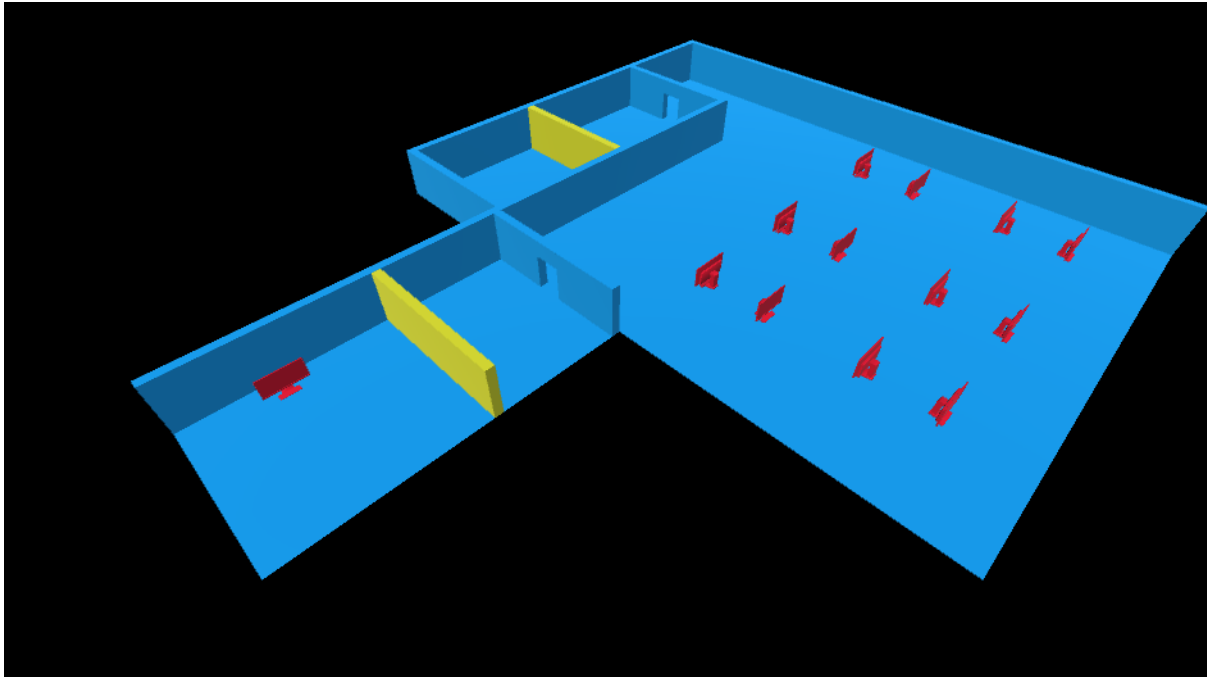


Figure 21. Controller Player View

Every time the players gain access to a new “access point” there is a random chance that they will be detected, at this stage the controller player must participate in a top-down shooting minigame where they must destroy the “viruses” that are attacking them. The controller player can move their character with either WASD or the arrow keys and they shoot using the mouse click and aim with the mouse.

4.4.2 Virtual Reality Player

For movement it was chosen to focus on teleportation in addition to a room scale rig. Room scale VR allows the user to move around in the real world and see their movements 1:1 in the virtual world. The levels in the game are significantly larger than the size of room scale however, so teleportation was needed. Support for more traditional continuous movement was added later after some testing. It was relevant in cases where the player was able to handle that movement style and wanted to improve their immersion.

Representing the VR player in the world was another challenge to tackle, the player could not be very realistic as it would stand out in the game and ruin their immersion, but the player needs some reference points to keep track of themselves in the virtual world.

The final part of implementing the controls was determining how the player would interact with elements in the game. This includes grabbable items, sockets and virtual buttons.

4.4.2.1. Movement and Tracking

Teleporting for the player is as simple as pointing at the ground and moving the thumb stick, when they release the thumb stick, they move to the desired point on the ground. They are only able to teleport to floors that support it, this allows for the players movement to be limited so they cannot teleport outside of the map. To show this point on the ground, a line is drawn from the players hand to the point, then a reticle is displayed on the floor showing the player where they will teleport to and in what direction they will be looking. The reticle is attached to the floor game object, this prevents the reticle from displaying when the player is not able to teleport, so they do not get confused.

For room scale movement, the player just walks around them in the real world. When the player reaches the boundary of their space, a boundary is displayed to them. The boundary is essential as it prevents them from walking into their wall in the real world.

Continuous movement works like any other movement in a game, the player moves the thumb stick in the direction they wish to travel, and they will move in that direction. The player has a collider on them so even while using continuous movement they are not be able to walk straight through walls or leave the map.

Other movement options for the player include snap turning and turning around completely. The player can turn by snapping the thumb stick in that direction rather than turning in the real world. This can be helpful for users who must deal with lots of wires coming from their headset. The number of degrees of the turn can be chosen by the player in the main menu as well as enabling or disabling these options. The player is also able to choose continuous or teleportation movement in the menu.

In terms of tracking, it was decided to not use a humanoid avatar for the player. The animations of the player character can be very off-putting in many situations like when tracking is lost for a moment. This would further lead to breaks in immersion. To mitigate this, it was chosen to use a simple controller shape for the player's hands. This technique of only showing the player's hands in the game is common among many VR games as it gives them the presence,

they need without entirely breaking their immersion. In addition to this, ‘tomato presence’ (38) is used. Tomato presence is a process where when the player grabs an item their hand becomes that item rather than showing their hand holding onto the item. It has been found that this technique improves player control and does not have an effect on immersion (38).

The movement of the hands are done using physics rather than just directly moving the hands via tracking. This allows the hands to apply force to objects in the scene, for instance pushing a pen across a desk. This is essential to allow the player to be able to use the physical buttons in the game. They can intuitively just press the button like in the real world rather than pressing a separate button on their controller.

4.4.2.2. Interactions

For grabbable items a collider is added, a grab script and a rigid body. The collider prevents the item from going through other objects and the rigid body allows the built in *Unity* physics to control certain aspects of the object. In this way they can have different mass and drag for example, this would make a heavy object travel slower than a light object. Figure 22 below shows what the script looks like in the editor. Various aspects of the interaction can be defined here.

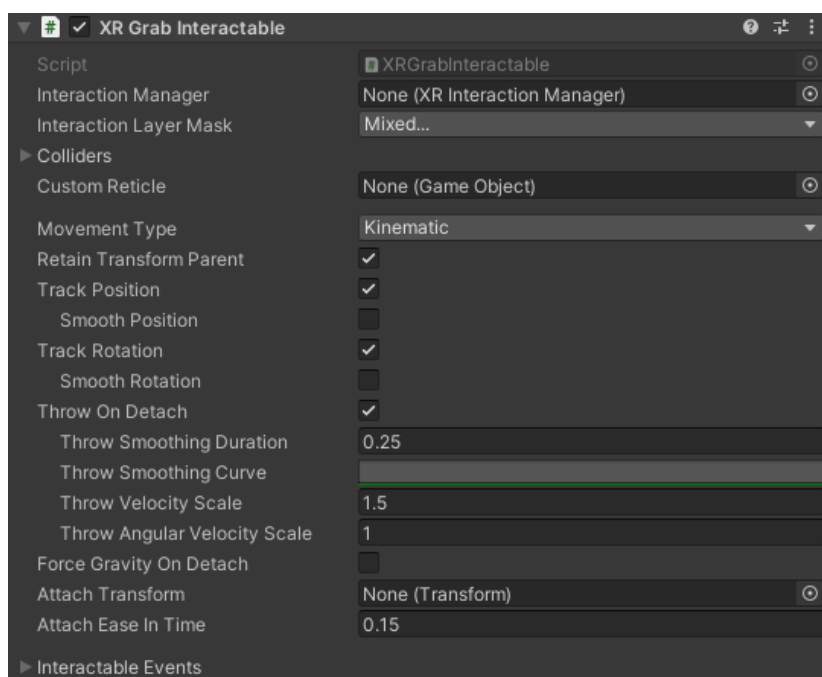


Figure 22. XR Grab Interactable Code as it appears in Unity

Sockets are areas in the world where a particular item can be attached, an example of this in the game would be when a USB is connected to a computer. In this example, a socket is added to the computer model. There is a sphere collider in the position where the USB can be plugged in, when the USB enters this collider the player can let go of the USB and it will attach to the computer. Also, while the player still holds the USB, the USB will appear in the position in green if it is the correct game object to attach to the socket or red if it is the incorrect object.

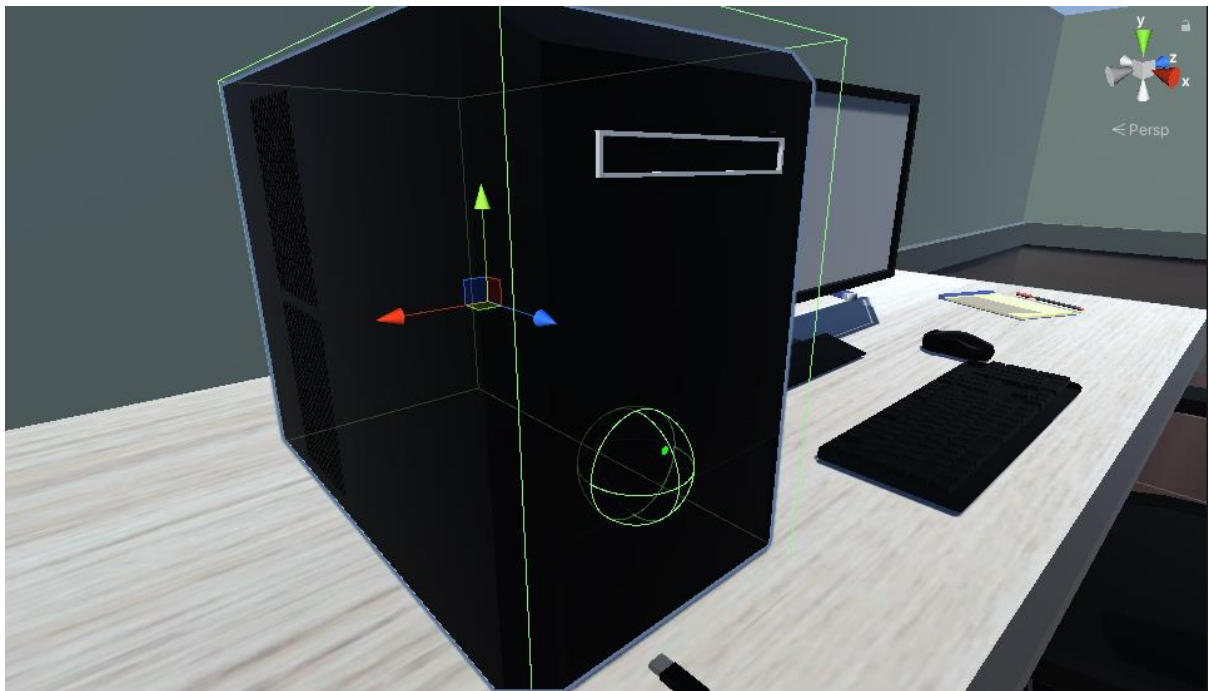


Figure 23. Colliders on Computer Asset

The USB is similar to the grabbable objects except it also interacts with a layer called USB key. The socket is only able to interact with objects on this layer in the game, so only a USB that can interact with that layer will attach. The socket itself is controlled by a socket interactor script; it appears like figure 24 below in the editor.

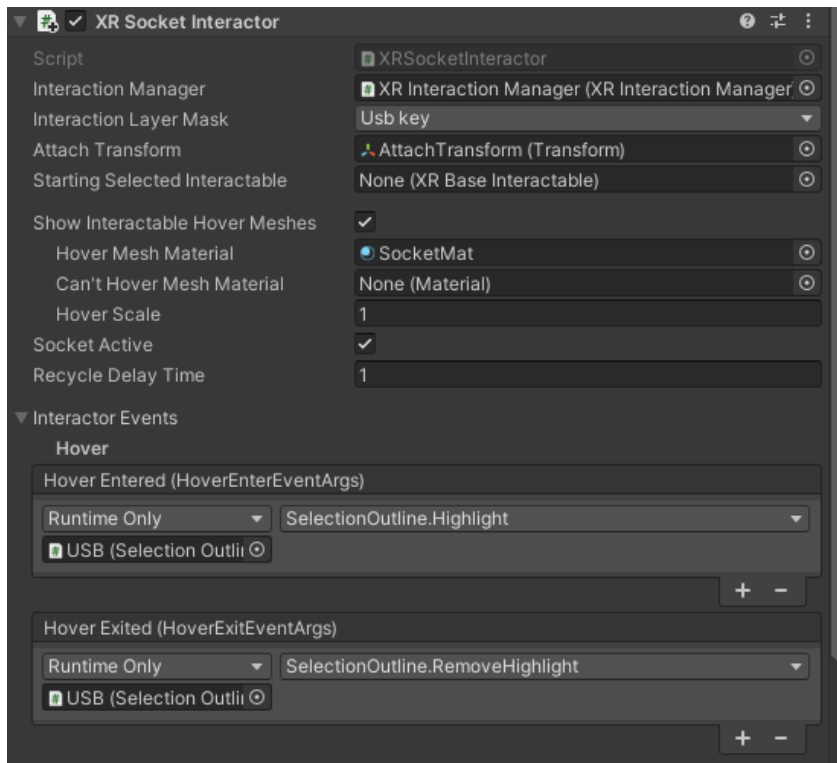


Figure 24. XR Socket Interactor Code as it appears in Unity

When the USB successfully attaches to the socket on the computer, it increases the controller player's ability to hack.

The physical buttons consist of 3 parts: the base, the physics and the button itself. The base is just a cylinder made in *Unity* with a convex mesh collider and a rigid body added to it. The collider is set to convex as it allows the button model to go through it to appear like it is being pressed. For the physics part of the button there are a few components. These include a box collider for the player to interact with, a rigid body to let it work with the physics in *Unity*, a configurable joint and a script controlling the threshold, dead zone and what happens when the button is pressed and released. The configurable joint is what allows the button to behave like a button, it limits the movement of the button to one direction so it can only move down and up. It also adds the spring behaviour to the button so that it springs back up after being pressed. The script controlling the logic of button is shown below in figure 25 as it appears in the editor. The actual physical button asset is another cylinder made in *Unity*; this is set as a child of the physics part of the button, so it is rendered without anything else attached to it.

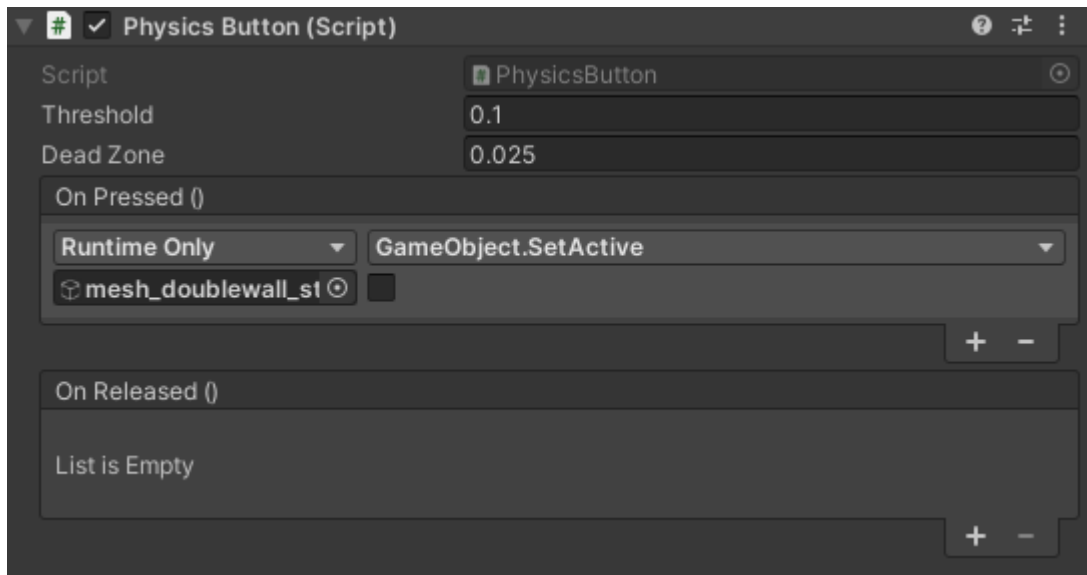


Figure 25. Physics Button Code as it appears in Unity

4.5. Bomb Defusal

At certain stages in the game, the VR player will be tasked with defusing bombs in the level. The controller player will be able to tell the VR player what they must do to diffuse the bomb, however they do not know what it looks like so they will rely on that information from the VR player.

The model for the bomb was created in *Blender* and the materials were created in *Unity*. The timer on the bomb is a *Unity* UI text element. A UI canvas is set as a child of the screen and the text element is a child of this canvas. A script is attached to the canvas, this script controls the countdown and updates the text accordingly. If the timer runs out, the bomb will explode. The figure below shows the script in the editor view.

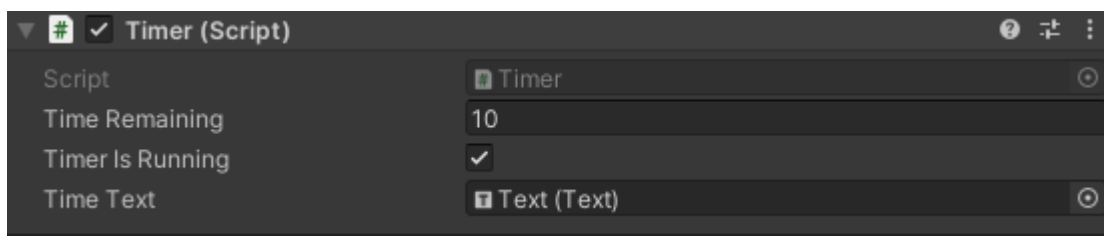


Figure 26. Timer Code as it appears in Unity

In addition to the *Blender* model and the text element, a wire is created in real time in *Unity*. The wire is set as a child of the bomb so that it moves with it in the game. The wire has a rigid

body, rope script and *Unity* line renderer attached to it. The rigid body allows it to be affected by physics, the rope script shown below in figure 27, instantiates the joints of the wire and sets the start (hook) and end (weight) points of the wire. The line renderer is drawn using the position of each link as a point in the line, resulting in one continuous line.

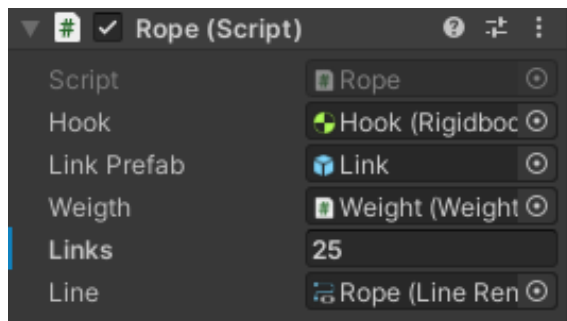


Figure 27. Wire Code as it appears in Unity

The hook is attached to the wire as a child, it has a rigid body and hinge joint attached to it. The hinge joint is what allows the wire to behave like a real-world wire, it is shown below in figure 28. Each link in the wire is set up the same as the hook except a script is attached to them that sends a message to the parent object (the overall wire) once it gets deleted from the scene. Each link can be interacted with by the VR player, and they can press to “cut” the wire, this results in the link being deleted from the scene. The message is sent which tells the bomb that the wire has been cut so the bomb can decide whether it should explode or stop the timer.

The start and end points of the wire are the only points that are constrained so when the wire is cut, it hangs and swings like a real wire rather than snapping to a “cut position” that is common in games. This further attention to realism is essential in a VR game like this as users can be a lot less forgiving when it comes to objects behaving realistically unlike in a traditional game.

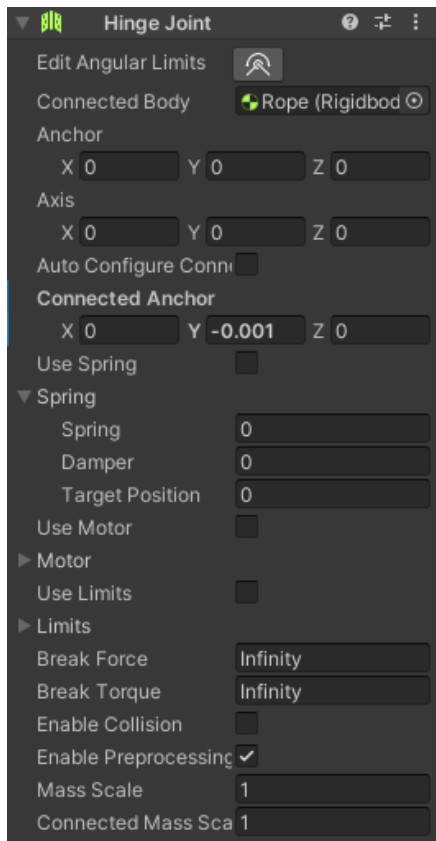


Figure 28. Unity Hinge Joint Settings

The end point (weight) is a separate element that is set as a child of the bomb but not the wire itself. It has a weight script and rigid body attached to it. The script simply attaches it to the end of the wire. The rigid body of the weight is set to kinematic; this prevents the physics in *Unity* from moving it. If the weight could move, then the wire would be swinging down off the bomb and when cut, one half of the wire would end up on the floor rather than both ends hanging from the bomb like they should be. Despite the weight not being moved by physics it is still possible to move it through script, so when the user picks up the bomb, the wire will move accordingly. The below figure 29 shows the final version of the bomb.



Figure 29. Final in-game model of the Bomb

4.6. Infringement Detected Mechanic

In order to be able to ‘hack’ certain objects in the game, the controller player needs an ‘access point’. The controller player gains an access point every time the VR player places a USB into a designated computer in the game. When this happens the controller player will be able to hack objects for information, open doors or reveal hidden passageways. Every time the players gain a new access point there is a random chance that their ‘infringement’ will be detected. When this happens the controller player will be moved to a new screen that is a top-down shooter where they must defeat the ‘viruses’ attacking them. There are two types of viruses that can attack the player a stationary turret type and a seeking type.

4.6.2. Turret Type

The turret type has two states, waiting and shooting. The turret spawns in a random position on the platform the minigame takes place on. It then stays in the waiting state until the player enters within its range, at this stage the turret will turn towards the player and start shooting. The behaviour tree for the turret is shown below.

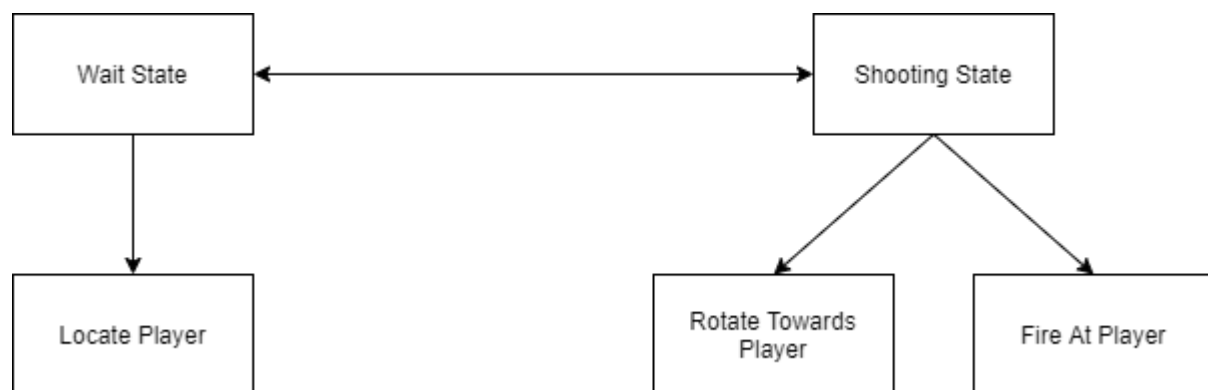


Figure 30. Turret Behaviour Tree

To turn towards the player, the turret takes the position of the player and subtracts its own position from this. It then rotates towards this new position.

To start shooting the turret uses a coroutine. An ordinary function in *Unity* would complete within one frame (39). If the shooting used an ordinary function it would work ok unless the player left the range of the turret and then entered again, in this case the shooting would have

to start from the beginning again. Doing it as part of a coroutine allows the shooting to pause between frames or mid frame and then resume immediately when required. Another benefit of the coroutine is that it allows the turret to shoot at a constant rate rather than fluctuating with the frames. The bullet used is instantiated at a point in front of the turret. A script on the bullet moves it forward toward the position of the player at the time of firing.

The turret fluctuates between these two states based on the position of the player. There is a sphere collider around the turret, this is set as a trigger. As it is a trigger it does not stop the player from entering but does notify the turret that the player has entered the collider. When the player enters the collider, it will start to shoot and when the player leaves the collider it will stop shooting.

4.6.2. Seeker Type

The seeker type of virus actively searches for the player. They stay in a wander state where they move to random positions around the playable space until it finds the player. When it finds the player, it will enter a pursuit state, in the pursuit state the virus will follow the player and shoot at them until it or the player is destroyed.

For the wander behaviour it just selects a number of waypoints and follows them until the player enters the range of the virus.

When the player enters the range of the virus, the pursuit behaviour will activate, and it will start to follow the player rather than random waypoints. The virus will then begin to shoot the player in the same way as the turret type. The behaviour tree for this type can be seen below.

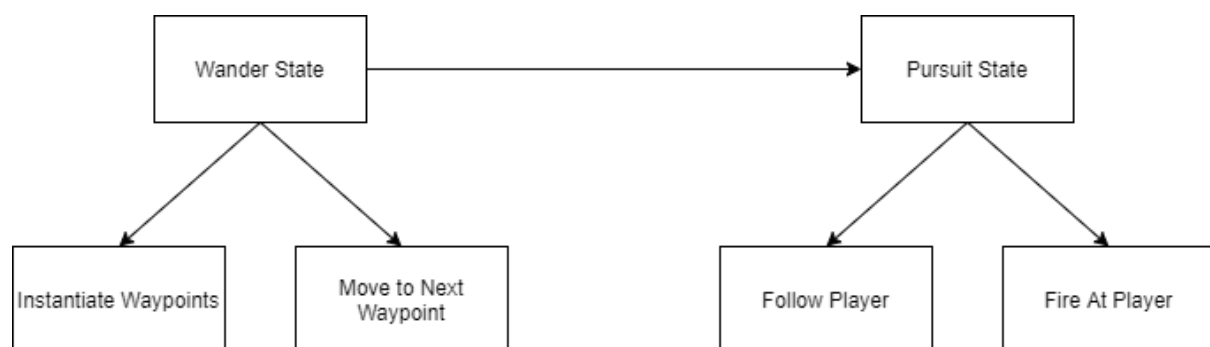


Figure 31. Seeker Behaviour Tree

4.7. Conclusions

The development for the system went well throughout. Scrapping the network element of the project at the late stage of January was not ideal but thanks to the research, design and planning done from the offset of the project it did not have a negative effect on the development of the project.

The final developed project was a platform on which to conduct testing and provided a compelling core game loop that emphasised clear communication between the players as an essential element.

The experience created for both the VR and controller players was different enough that it allowed players to replay in a different role without feeling repetitive.

A compromise had to be made in terms of the art style of the game, making it more abstract as a large team would be needed to create convincing assets that were photorealistic. This compromise did not generate any negative comments from testers up to this stage, but it may do so at some point in the future.

5. Testing and Evaluation

5.1. Introduction

This chapter will discuss the testing and evaluation of the system. The testing was focused on performing user testing and Blackbox testing both while developing the game and on the final build of the game. The evaluation will include testimony from a variety of potential users based on their level of immersion in the game.

5.2. Testing

The Project was continuously tested throughout its entire life cycle. Due to the on-going situation with Covid-19, user testing was operated remotely. The service Parsec (desktop streaming software) was used to allow testers to access up-to-date versions of the game without the need to download. The use of Parsec was dropped from testing during development however as it proved to induce motion sickness due to poor internet connections. The VR part of the game is more difficult to test as potential testers would have to own their own headset or do the testing non-remotely. To account for this shortcoming in VR testing, regular demos will need to be recorded and shown to testers that cannot participate, as a simpler test.

Backing up and committing the project consistently using GIT version control ensured that any changes could be rolled back when there were major errors.

Blackbox testing was used as it provides a different angle when testing the project. The tester does not know about the internal structure/ design/ implementation of the item being tested. The black box testing method was used to find errors in areas such as: Interface errors, performance errors, incorrect functions, and initialization errors.

In addition to this, users were asked to follow certain scenarios to see if they could navigate the interface efficiently.

It was decided that a mix of both directed testing and black box testing would be an adequate way to test the functionality of the project. Manual testing took place too wherein the developer played the end user and used the features to ensure correct behaviour.

Evaluation of this project will be done by the developer and potential users.

Test Plan

Table 1 Test Plan

Test No.	Test Description	Expected Outcome	Pass?
1	Does the game load when the icon is selected?	The game will load and bring the user to the main menu scene.	
2	Does the game close when the quit button is pressed?	The game will shut down correctly when quit is pressed.	
3	Do the players see different screens?	The VR user is in the level, the controller player is using the controller player interface.	
4	Can a User select a level to start?	The user can select a level from the menu and the level will load.	
6	Can the VR user interact with objects?	User can interact and pick up objects in the game world.	
7	Can the controller user interact with the game world?	The VR user can see the effects of the Controller user on the world.	
8	User completes level.	Both users are notified of completion of the level and the next level is loaded.	

5.3. Evaluation

Evaluation of this system is equally as important as testing. The reason for this is that user experience is one of the main complexities of the project and the level of immersion needs to be quantified.

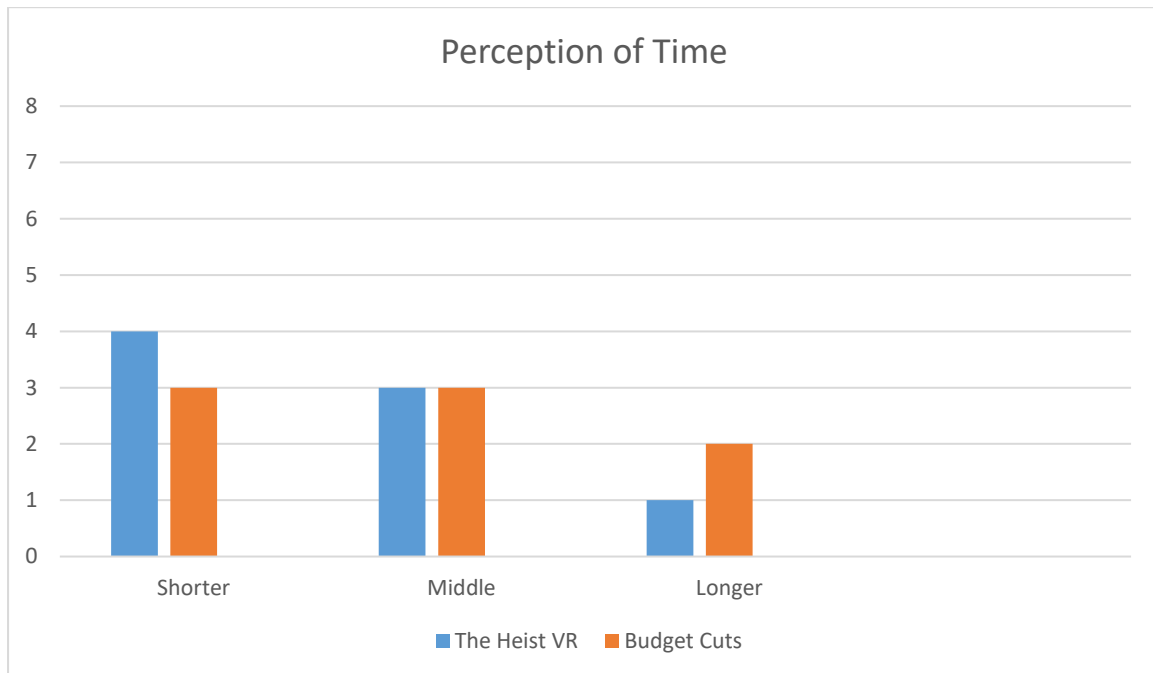
On the surface, the game needs to be as easy to use as possible and intuitive, even though underneath this there is a lot of complex code. As previously mentioned, the usability of the system being intuitive and instinctive was of high priority through the entire development process. Having the system be evaluated by potential users ensured that the system usability is of high quality and as close to an industry level proof of concept (demonstration) as possible in the time frame of the project.

In terms of immersion, this can be measured objectively (task completion time, eye movements) and subjectively (questionnaire) in a number of ways (40). For this project, eye movements was not a viable option as the VR player's eyes are obscured by the headset. A questionnaire was formulated to ask the users as objective questions as possible.

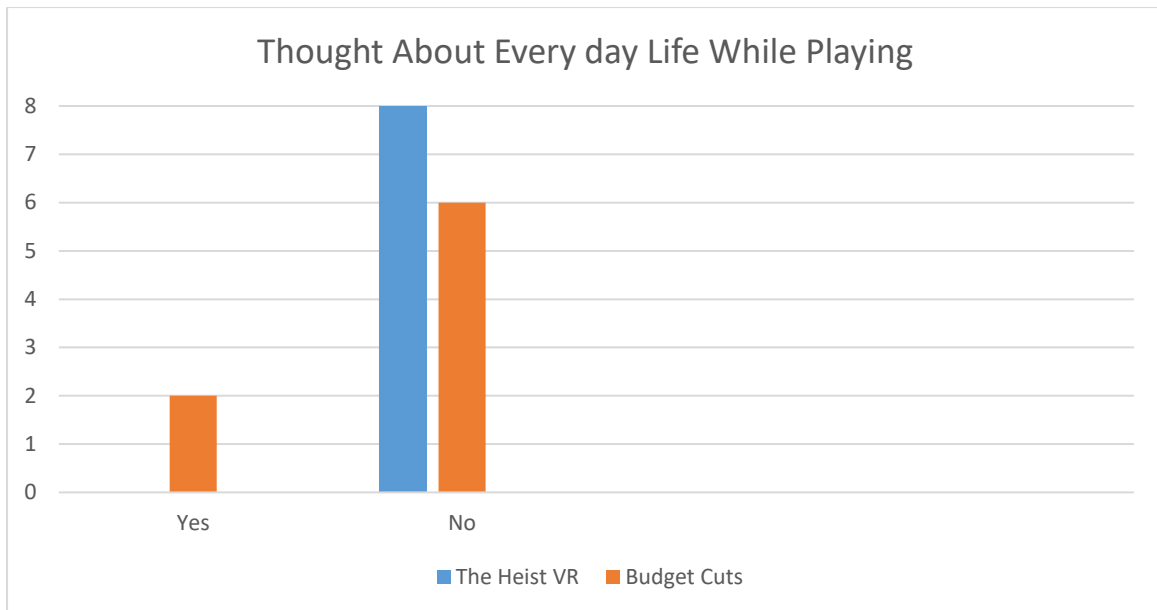
Immersion is achieved through concentration, feeling a connection to the virtual world and feeling excited by the pace of the game (41). The questions posed to the players followed this formula. The time taken for each play session was recorded and the first question posed to users was how long they spent playing the game. If the answer was shorter than the actual time that elapsed, then the level of immersion gained a point. If the answer were longer than the actual time this would suggest that the user was bored and not immersed in the world, the immersion level would lose a point. If the time was within a minute, then this would be considered as a margin of error and the level of immersion would neither gain nor lose a point. The next question asked the user if they thought about everyday life while playing. A no reply would give a point to immersion and a yes reply took a point away. The penultimate question asked the user if they felt viscerally involved in the game. Like before a yes reply gave a point to immersion and a no reply took away a point. The final question asked the user if they became aware of their real-world surroundings at any stage while playing. The options were often, sometimes and never. An answer of often, took a point away from immersion. Sometimes neither took a point away nor added a point to immersion. A reply of never gave a point to immersion.

To come to a final conclusion on immersion the project was compared to the game *Budget Cuts* (37). *Budget Cuts* is a game that requires similar tasks by the VR player and has a similar art style and premise.

5.4. Results from the Questionnaire

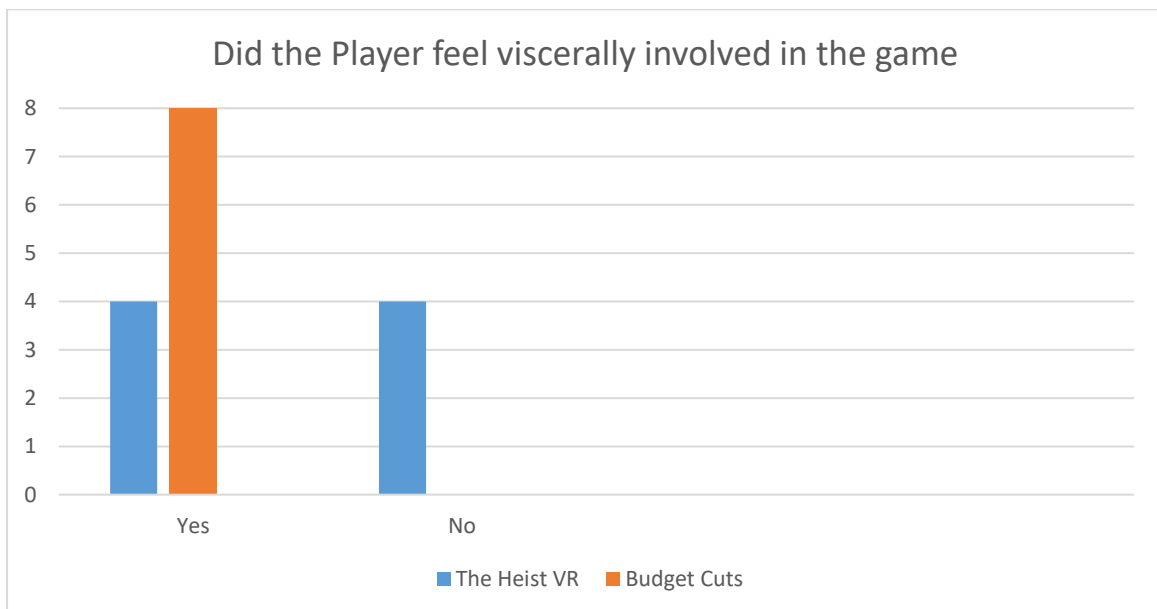


The results from the first question about the user's perception of time while in the game. The two games vary at the extremes of the distribution but are the same towards the middle.

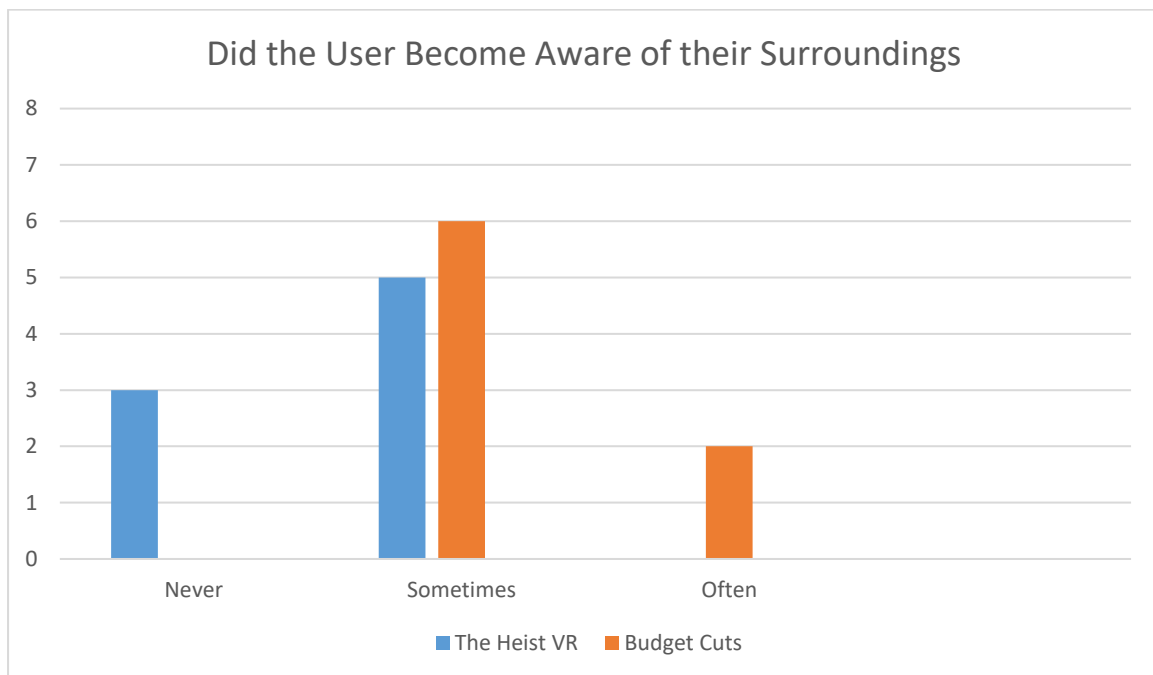


The results from the second question, showed those who played the project did not think about their everyday life. It is possible that having the distraction of an additional player in the game was the cause of this rather than being totally immersed in the game.

Budget Cuts managed to distract most players from their everyday lives, but 2 players reported thinking about their everyday life.

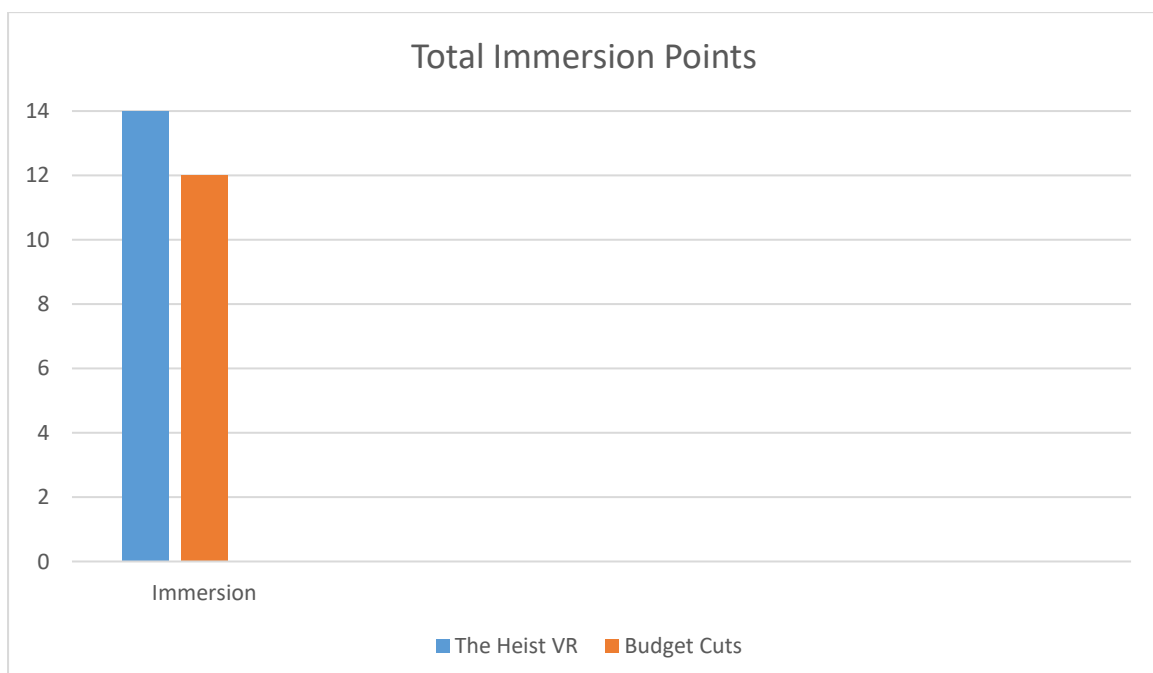


In this question, *Budget Cuts* got the most points for immersion with all players feeling viscerally involved in the game. *Budget Cuts* has mild combat mechanics, this may have added to the visceral feeling of the players. In *The Heist VR*, the VR player has no combat options, so the only visceral feeling comes from manipulating objects in the environment.



Mostly the users did not become aware of their real-world surroundings. As the project trends towards not becoming aware of their surroundings you can conclude that inviting the real-world into the game may have had a positive effect on immersion in this case.

Another reason for this disparity may be the combat mechanics of *Budget Cuts*. The combat in the game requires the player to throw objects, in some cases, players would intend to throw an object and end up hitting something in the real world with their hand.



5.5. Conclusion

This chapter reviewed the testing and evaluation of the system. The testing consisted of performing user testing and Blackbox testing both while developing the game and on completion. The evaluation included evaluation from a variety of potential users who were asked questions based on their level of immersion in the game.

Due to the pandemic, the number of users tested was lower than would be preferable. This would suggest that the results of the testing are inconclusive up to this stage and future work would have to be done to ensure a conclusive result. From the results taken, it would suggest that the difference in immersion between the two games is minimal. However, it is shown that by including another player in the real world to play with, does have a positive effect on the level of immersion.

The questionnaire for immersion showed how each game was lacking in certain aspects of immersion as well as areas where they excelled in this ambition.

6. Conclusions and Future Work

6.1. Introduction

This chapter will discuss any issues encountered thus far. All conclusions from development and the results of the project will be examined. Finally, ideas for work to be done into the future will be discussed.

6.2. Issues

There was a number of challenges that were involved in developing the project.

A lack of experience and knowledge in using *Blender* was a challenge that had to be overcome. An online course in *Blender* was completed and extra research was put into that area especially when working with *Unity*. The course was done on LinkedIn Learning thanks to the university's optional LinkedIn Learning account. This was completed after the interim stage and provided a good foundation in 3D modelling for the project.

Issues with the custom network caused certain development plans to be altered in the project. The final game is better because of these changes and performance has been improved (increased frames per second).

The Coronavirus pandemic caused numerous issues throughout the lifetime of the project. This included getting VR equipment and having a sufficient number of testers. Plans for in person testing had to be scrapped due to restrictions. It was possible to overcome by contacting fellow students and other VR users to test the project on their own devices.

6.3. Conclusions

The project was set up to see if the level of immersion of a VR player could be improved using asymmetric multiplayer. From the results gained, it can be seen that it is possible to improve immersion using this game design technique. The findings show consistent increased immersion in all aspects apart from visceral gameplay. This is not deemed to be fatal to the project's conclusions.

Given the difference in standard between *Budget cuts* a commercially developed VR game and *The Heist VR* a game created specifically for this project under severe time constraints and therefore lacking depth, the visceral gameplay results could be expected to be overturned and further research in this area is recommended.

The planning done throughout the project proved to be essential to allow the project to adapt to unexpected challenges. No major setbacks were encountered thanks to this fact.

Working on each aspect of the project proved to be interesting and rewarding. Working with 3D modelling software started off incredibly slow and frustrating but with time and practice it became easier to create convincing models and the final game shows this. The *Unity XR Interaction Toolkit* was in preview for the entire duration of the project and thus was difficult to work with at times as documentation was outdated, and deprecated functions were still included in the SDK. This never caused any big problem for the project however, and it still is the most supported VR SDK for *Unity*. Using this SDK allowed the project to be ported to other headsets and this was crucial when it came to testing as no on-site testing was allowed due to the pandemic. Working with game and level design was challenging, however the research done into similar games and game design as a whole was sufficient to develop a compelling core game loop for the players and a world that was easy to navigate.

The goals of the project were ultimately met. An immersive VR experience was created with social interaction as a core game mechanic. The immersion was tested in an objectively as possible way, and it is shown that this style of game can have a positive effect on immersion for the player.

6.4. Future Work

It is recommended that tweaks could be made, and features could be added so that the gameplay differential could be minimised to confirm the expectation that results are not fatally impacted by the gameplay.

Given the limited sample size of players, more participants would have to be used for any future work. With an increased sample population of players, more confidence could be attributed to the findings.

Another recommendation of future work is in-person testing which could not occur as a result of the pandemic.

The final point would be to compare the project to more VR games that have similar mechanics and even games that are completely different, to see how different mechanics and systems can affect immersion.

Bibliography

1. AR and VR Headsets Will See Shipments Decline in the Near Term Due to COVID-19, But Long-term Outlook Is Positive, According to IDC [Internet]. IDC: The premier global market intelligence company. [cited 2020 Dec 11]. Available from: <https://www.idc.com/getdoc.jsp?containerId=prUS46143720>
2. Slater M, Steed A. A Virtual Presence Counter. *Presence*. 2000 Oct 1;9:413–34.
3. Sylvester T. *Designing Games: A Guide to Engineering Experiences*. O'Reilly Media, Inc.; 2013. 416 p.
4. Crecente B. Nintendo's Fils-Aime: Current state of VR isn't fun - Polygon [Internet]. [cited 2020 Nov 25]. Available from: <https://www.polygon.com/2015/6/18/8803127/nintendos-fils-aime-current-state-of-vr-isnt-fun>
5. VRChat. VRChat [Internet]. VRChat; 2017. Available from: <https://hello.vrchat.com/>
6. Linden Lab. Second Life [Internet]. Linden Lab; 2003. Available from: <https://secondlife.com/>
7. Vankrupt Games. Pavlov VR. Vankrupt Games; 2017.
8. Official Site | Second Life - Virtual Worlds, Virtual Reality, VR, Avatars, Free 3D Chat [Internet]. [cited 2020 Nov 25]. Available from: <https://secondlife.com/>
9. Turtle Rock Studios. Evolve [Internet]. 2K Games; 2015. Available from: <https://2k.com/en-US/game/evolve/>
10. Hecker C, Cimino J. SpyParty [Internet]. Chris Hecker; 2018. Available from: <http://www.spyparty.com/>
11. Steel Crate Games. Keep Talking and Nobody Explodes - Defuse a bomb with your friends. [Internet]. Steel Crate Games; 2015 [cited 2020 Nov 25]. Available from: <https://keeptalkinggame.com/>
12. Llamas S. Will the Oculus Quest be the answer to VR's prayers? [Internet]. SuperData, a Nielsen Company. [cited 2020 Nov 25]. Available from: <https://www.superdataresearch.com/blog/will-the-oculus-quest-be-the-answer-to-vrs-prayers>
13. Team Future LLC. Black Hat Cooperative [Internet]. Team Future LLC; 2016. Available from: <https://www.teamfuturegames.com/>
14. Japan Studio. The Playroom VR [Internet]. Sony Interactive Entertainment; 2016. Available from: <https://www.playstation.com/en-ie/games/the-playroom-vr-ps4/>

15. Team VRGC. Best VR Party Games | vrgamecritic [Internet]. [cited 2020 Nov 25]. Available from: <https://vrgamecritic.com/article/great-vr-party-games-that-will-make-any-game-night>
16. Team Panoptes. Panoptic [Internet]. Team Panoptes; 2020. Available from: <https://panopticgame.com/>
17. Valve. Half-Life: Alyx [Internet]. Valve Corporation; 2020. Available from: <https://www.half-life.com/en/alyx/>
18. SIE London Studio. Blood & Truth [Internet]. Sony Interactive Entertainment; 2019. Available from: <https://www.playstation.com/en-ie/games/blood-and-truth/>
19. Heeter C. Being There: The Subjective Experience of Presence [Internet]. [cited 2020 Nov 25]. Available from: <http://commtechlab.msu.edu/randd/research/beingthere.html>
20. Cairns P, Cox AL, Day M, Martin H, Perryman T. Who but not where: The effect of social play on immersion in digital games. *Int J Hum-Comput Stud*. 2013 Nov;71(11):1069–77.
21. Liszio S, Masuch M. Designing Shared Virtual Reality Gaming Experiences in Local Multi-platform Games. In: Wallner G, Kriglstein S, Hlavacs H, Malaka R, Lugmayr A, Yang H-S, editors. *Entertainment Computing - ICEC 2016*. Cham: Springer International Publishing; 2016. p. 235–40. (Lecture Notes in Computer Science).
22. Alice-Bonasio-VR-Consultancy-MR-Tom-Atkinson-Tech-Trends-Review-AR-Mixed-Virtual-Reality-Augmented-Dell-visor-headset-hand-controllers-microsoft-09-848x400.jpg (848×400) [Internet]. [cited 2021 Apr 6]. Available from: <https://techtrends.tech/wp-content/uploads/2018/04/Alice-Bonasio-VR-Consultancy-MR-Tom-Atkinson-Tech-Trends-Review-AR-Mixed-Virtual-Reality-Augmented-Dell-visor-headset-hand-controllers-microsoft-09-848x400.jpg>
23. Oculus | VR Headsets & Equipment [Internet]. [cited 2021 Apr 6]. Available from: <https://www.oculus.com/>
24. Roosendaal T. Blender [Internet]. The Blender Foundation; 1998. Available from: <https://www.blender.org/>
25. Autodesk. Autodesk 3Ds Max [Internet]. Autodesk; 2020. Available from: <https://www.autodesk.eu/products/3ds-max/overview?plc=3DSMAX&term=1-YEAR&support=ADVANCED&quantity=1>
26. Autodesk, Alias Systems Corporation. Autodesk Maya [Internet]. Autodesk; Available from: <https://www.autodesk.eu/products/maya/overview>
27. 3D Modelling for Unity: The Complete Guide | Game-Ace [Internet]. 2019 [cited 2020 Nov 26]. Available from: <https://game-ace.com/blog/3d-modeling-for-unity/>
28. Whyte S. ‘Blood & Truth’: Lessons Learned Making a VR Action Movie [Internet]. XR Developer Conference; 2019 Oct 14; San Francisco, Fort Mason Festival Pavillion. Available from: <https://www.gdcvault.com/play/1026555/-Blood-Truth-Lessons-Learned>

29. Valve Corporation. Half Life: Alyx Developer Commentary. 2020.
30. Cherni H, Métayer N, Souliman N. Literature review of locomotion techniques in virtual reality. *Int J Virtual Real*. 2020 Mar 27;20(1):1–20.
31. Photon Unity 3D Networking Framework SDKs and Game Backend | Photon Engine [Internet]. [cited 2020 Dec 14]. Available from: <https://www.photonengine.com/pun>
32. Mirror Networking – Open Source Networking for Unity [Internet]. [cited 2020 Dec 14]. Available from: <https://mirror-networking.com/>
33. Flanagan P. Enhancing Immersion in Virtual Reality [Internet]. Dublin Institute of Technology; 2017. Available from: <https://ditlib.dit.ie/articles/3766229.2971/1.PDF>
34. Nevin K. Dragon Simulator [Internet]. Dublin Institute of Technology; 2018. Available from: <https://ditlib.dit.ie/articles/4187235.3560/1.PDF>
35. Why should you use a feature-driven development? [Internet]. [cited 2021 Apr 6]. Available from: <https://aist.global/en/use-a-feature-driven-development>
36. Labs O. Job Simulator [Internet]. Owlchemy Labs; 2016. Available from: <https://jobsimulatorgame.com/>
37. Budget Cuts [Internet]. Neat Corporation; 2018. (Budget Cuts). Available from: <https://neatcorporation.com/>
38. How to Hands: A Developer Deep Dive on Hand Tracking in Vacation Simulator [Internet]. 2020 [cited 2021 Mar 30]. Available from: <https://owlchemylabs.com/how-to-hands-a-developer-deep-dive-on-hand-tracking-in-vacation-simulator/>
39. Unity - Manual: Coroutines [Internet]. [cited 2021 Mar 31]. Available from: <https://docs.unity3d.com/Manual/Coroutines.html>
40. Jennett C, Cox AL, Cairns P, Dhoparee S, Epps A, Tijs T, et al. Measuring and defining the experience of immersion in games. *Int J Hum-Comput Stud*. 2008 Sep 1;66(9):641–61.
41. Sweetser P, Wyeth P. GameFlow: a model for evaluating player enjoyment in games. *Comput Entertain*. 2005 Jul;3(3):3–3.
42. Hettinger LJ, Riccio GE. Visually Induced Motion Sickness in Virtual Environments. *Presence Teleoperators Virtual Environ*. 1992 Jan 1;1(3):306–10.

Appendix

Game Design Document starts on next page.



<< THE HEIST VR >>

GAME DESIGN DOCUMENT
BY PHILIP TOOLAN

C:\Users\TheHeistVR\GeneralVision

Logline:

In this asymmetric co-op VR game you are a team of corporate agents tasked with stealing your competitors prototype. One must sneak around the offices and facilities while the other guides them and ensures they do not get caught. Will you complete your task without being detected?

Running:

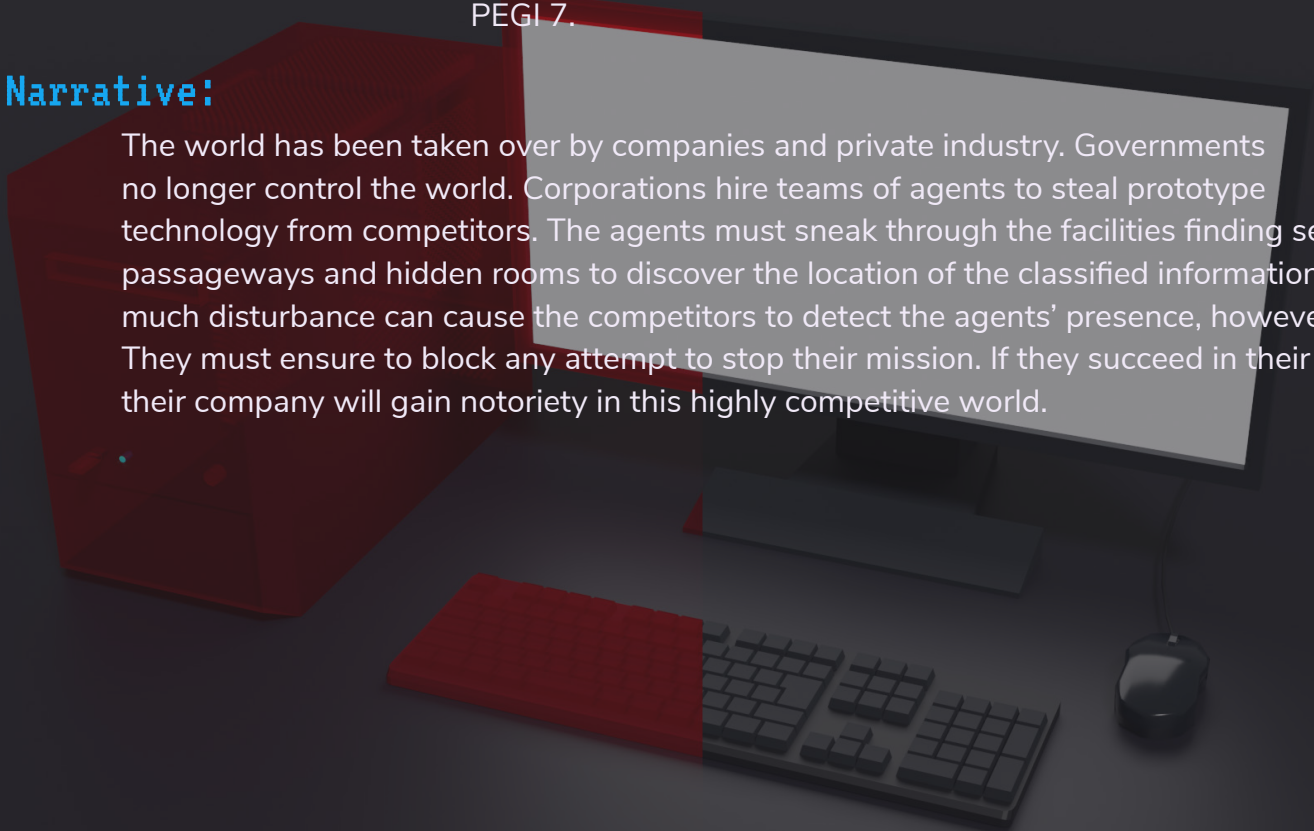
Genre Tactical stealth, co-op.

Platforms Developed with Dell Visor VR. Compatible with Oculus rift, Oculus Quest 2, HTC Vive, Valve Index. PC.

Target Ages 15-26 looking for an immersive experience.
PEGI 7.

Narrative:

The world has been taken over by companies and private industry. Governments no longer control the world. Corporations hire teams of agents to steal prototype technology from competitors. The agents must sneak through the facilities finding secret passageways and hidden rooms to discover the location of the classified information. Too much disturbance can cause the competitors to detect the agents' presence, however. They must ensure to block any attempt to stop their mission. If they succeed in their heist, their company will gain notoriety in this highly competitive world.



C:\Users\TheHeistVR\CompetitiveAnalysis

Inspiration:

BlackHat Cooperative	Asymmetric video game
Keep Talking and Nobody Explodes	Asymmetric video game
Job Simulator	VR video game

Competitors:

BlackHat Cooperative	Asymmetric video game
Panoptic	Asymmetric video game
I Expect you to Die	VR video game

What makes the game different?

Both players have their own game loop, they need to receive help from the other player rather than help flowing in one direction only.

C:\Users\TheHeistVR\Playability

Objective

Steal the prototype without being detected.

Mechanics

VR	Movement through teleporting or continuous movement, room scale tracking. Interact with objects in the scene using the grip button. Press buttons in game by moving their hand.
Controller	Click on objects to get details. In hacking minigame, move with WASD or arrow keys and shoot with the mouse.

Obstacles

Click on objects to get details. In hacking minigame, move with WASD or arrow keys and shoot with the mouse.

Resources

The controller player must use access points to hack certain objects in the game, they receive one of these points every time the VR player places a USB into a specific computer in the game.

