# Final Report

## Title Page

## Acknowledgements

## Abstract

Contents

[Final Report 1](#_Toc478555433)

[Title Page 1](#_Toc478555434)

[Acknowledgements 2](#_Toc478555435)

[Abstract 2](#_Toc478555436)

[Main body 4](#_Toc478555437)

[Introduction 4](#_Toc478555438)

[Background 4](#_Toc478555439)

[Objectives 6](#_Toc478555440)

[Deliverables 6](#_Toc478555441)

[Literature review (if applicable) 6](#_Toc478555442)

[Method of approach 6](#_Toc478555443)

[Legal, social, ethical, and professional issues 6](#_Toc478555444)

[Project management 6](#_Toc478555445)

[Stage 1 6](#_Toc478555446)

[Stage 2 6](#_Toc478555447)

[Stage 3 6](#_Toc478555448)

[Stage 4 6](#_Toc478555449)

[Project post-mortem 6](#_Toc478555450)

[Conclusions 6](#_Toc478555451)

[Statement of word count 6](#_Toc478555452)

[Reference List 6](#_Toc478555453)

[Bibliography 6](#_Toc478555454)

[Appendices 6](#_Toc478555455)

[User Guide 6](#_Toc478555456)

[Project Management Artefacts 6](#_Toc478555457)

[Other materials (UMLs, designs, test results) 6](#_Toc478555458)

## Main body

### Introduction

The purpose of this project was to create a Virtual Reality game made using Unity3D, intended for use with the HTC Vive HMD (Head mounted display).

Although there was never a definitive intended client, there are many potential ones. Small innovative indie games can often end up on digital distribution platforms such as Steam after being noticed by online communities such as Steam Greenlight. In this case the intended client is any consumer interested in pc gaming, particularly those who are interested in small-scale, innovative games.

### Background and Project Objectives

**Initial idea for the project**

The initial plan for the project was a single-player VR puzzle game. After exploring the best potential puzzle-like applications for VR, a plan was drawn up for a “Factory Worker Simulation game” . The player would stand in front of a conveyor belt with different items moving past them on the conveyor. The player would have to do something with the items e.g. sort them, throw specific ones away etc. However, after about a week this idea was scrapped. (An image of the prototype can be found in the appendices). The reason for this was because the idea could not easily be expanded on to produce a full-bodied game. This was not a wasted experience though, as it highlighted that more planning was necessary to ensure that an idea be fully fleshed out before any development starts.

More planning let to a new idea – a game consisting of a set of “mini-games”. Inspiration was taken from Valve’s “The Lab”, described as a “compilation… of room-scale VR experiments”[[1]](#footnote-1). With high-end VR being (relatively speaking) in its infancy, these sorts of games give the best introductory experience to virtual-reality due to their varied nature and intuitive mechanics. The idea was that the player would start in a hub room and could travel to whichever mini-game he chose to play/experience, and then travel back to the hub room to select a new game.

To solidify the project’s objectives a Project Initiation Document (PID) was drafted (available in the appendices) which gave an overview of the project details. The aims for the project were specified in terms of its scope. This was split into things that should definitely be achieved, things likely to be achieved, and things that may be achieved if there is enough time. The PID proposed a “puzzle based game”, with the ability to “pick up and interact with objects”. It also specified at least “20” short puzzle based levels, 15 3D modelled assets and at least 10 hand recorded sounds. In hindsight, it appears that most of these aims were met, whilst some that ended up becoming no longer entirely relevant were re-adjusted.

Talk about choosing PlaySpace VR later and trying to make it look like a play room

**Choosing a platform**

One of the big initial steps in planning the project was to choose what platform it would be built for. I had to decide between a mobile or non-mobile platform. In the last decade there has been a “rise of smartphones with high-density displays and 3D graphics capabilities”.[[2]](#footnote-2) This makes them extremely practical as virtual reality devices. Google have already taken advantage of this by launching the ‘Google Cardboard’, bringing “immersive experiences to everyone in a simple and affordable way”[[3]](#footnote-3). Samsung followed suit soon after by releasing the ‘Gear VR’. Both these products are simply devices to hold your phone comfortably in front of your eyes. However, they simply do not have the processing capabilities to provide a smooth experience in most cases.

[[4]](#footnote-4)

In addition to the surge in cheap, mobile VR devices and applications, there has also been huge growth in VR head-mounted displays (HMDs). Three main competitors corner this market at present: Sony with the PlayStation VR, Oculus (now owned by Facebook) with the Oculus Rift, and HTC with the HTC Vive. These devices offer unparalleled performance for VR experiences due mainly to the quality of the displays. For example, both the Rift and Vive offer “two OLED panels boasting a combined 2,160x1200.” This means that “each eye gets its own 1080 x 1200 display.”[[5]](#footnote-5) As the goal of VR is to convince the brain that you really are in the virtual space being simulated, it is essential to have displays that offer a resolution as close to that of the human eye as possible. At present this is the main drawback of mobile VR; it simply does not offer a good enough resolution to make the experience completely convincing or immersive. After conducting this research, I settled on a non-mobile HMD as my target platform – specifically the HTC Vive because at the time only the HTC Vive had support for Unity3D, via an asset store plug-in.

### Personal Objectives

The predominant objectives whilst undertaking this project were to develop skills relating to Game development. These include but are not limited to Game Design, asset creation/3D modelling and proficiency in specific software such as Unity3D and Blender. In addition, carrying out a project of this scale as a solo developer from start to finish provided invaluable experience in development methodologies, effective versioning using GIT, bug tracking and time management.

### Deliverables

The main deliverable for this project was a Unity executable. However I also gathered some requirements information, produced a Gantt Chart to plan my work, took screenshots and video footage throughout development and drew designs at the start of the process. Together these provide a clear story of how the project progressed.

### Literature review (if applicable)

### Project Management / Method of approach

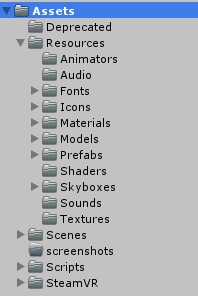
During this project, I utilised several software tools. The main three were Unity (for game development), Blender (for asset creation/modelling), and Git (for versioning).

A good, well-thought out approach can save a huge amount of time during a large project. Unity, Blender and Git all have ways to speed up and optimise the development process, especially when iterating between all three. For example, the developer set up Blender to save all assets straight into the Unity asset directory (under a subdirectory named “models”). This meant that when a blender file was saved, it would automatically update the Unity asset, keeping everything in-sync.

**Unity**

In terms of Software, Unity was chosen for the game development. This was down to several reasons. Firstly, the developer already had experience creating other, smaller, games with it. Secondly, at the time of the start of the project, the only other viable alternative (Unreal Engine), did not yet have any VR support.

Unity was for the most part a pleasure to work with. It provides organisational and structural tools which ensure as the project grows it does not become overwhelming or confusing, provided you put in the effort to keep things organised. For example, organising all assets into a hierarchical format can keep things streamlined and easy to navigate.

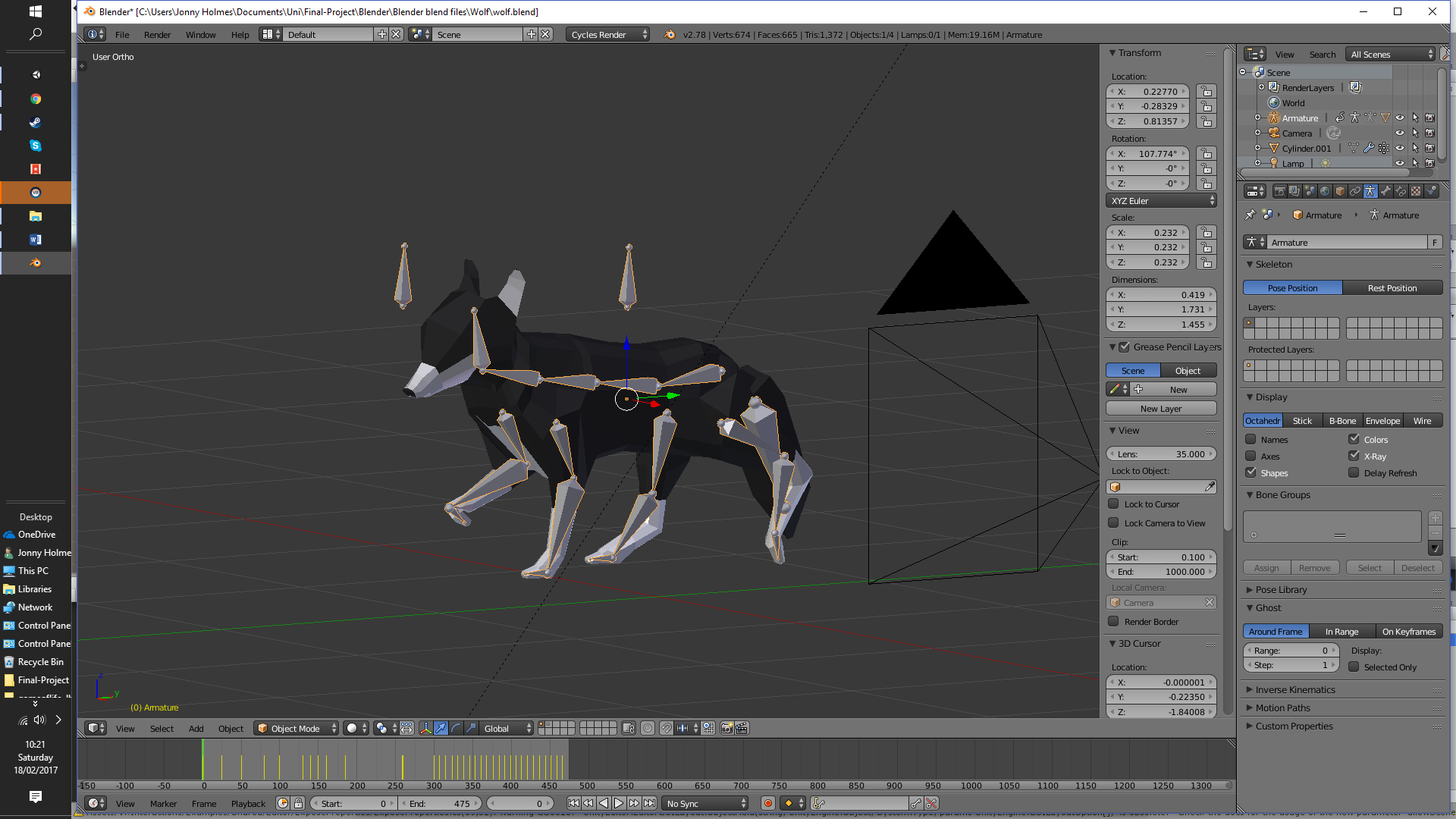


Another benefit of Unity is access to the asset store, where you can download third party assets. As the developer wanted to keep the project as independent as possible he refrained from using this, but it did provide the SteamVR Unity plug-in and the Skyboxes that were used in the project. Unity also makes it very easy to deploy to multiple platforms, which means that if the intended platform changed at any point in the development process it would make this transition much more manageable than other development tools.

Within Unity I set my default external scripting tool as MonoDevelop. MonoDevelop is a lightweight IDE used mainly for scripting. It includes all the essential features such as automatic code completion, source control and a GUI. However, the developer had problems with MonoDevelop crashing halfway through development and eventually switched to Visual Studio, which soon proved to be a more robust and stable IDE.

**Blender**

Blender was used for modelling / asset creation. Although Blender can be intimidating or seem chaotic to some developers, when focusing on a particular set of features the workflow becomes quicker and easier over time. It provided the developer with a diverse set of tools to model, rig and animate as he needed.



**Version control**

Git was used throughout the project as the version control software of choice. Instead of using the standalone GitHub software, GitBash was used to track, commit and push files from the local to the remote repository.

A commit was made almost daily during work periods, however when the project work required balancing with other commitments this number dropped to once every few days. This frequent backup ensured that if necessary the project could be rolled back to a previous commit, reducing risk of data loss through hardware failure or other means. Roll-backs were done on multiple occasions and potentially saved the project days of work. For example, when updating the version of the SteamVR plug-in for Unity, some bugs became present, probably because the update was new and not completely stable. Even after rolling back the plug-in version the bugs remained. It seemed the only solution was to revert back to a previous commit and wait for the bugs to be fixed before attempting the update again. Sure enough, after updating a week or so later no bugs emerged.

A few problems arose with Git throughout the project duration, which were documented in highlight reviews. One problem occurred because GitHub will not allow free users to upload any file over 100MB. Some of the project’s Blender files were above the limit. This problem was solved in two ways. Firstly, by compressing the Blender files by default to reduce their size; secondly by adding the Unity project metadata folder to the repository’s ‘gitignore’ list. This did not reduce the integrity of the backup as this meta-data is generated every time the project is launched in Unity and so can be excluded from versioning.

**Development Methodologies**

As software development processes work much better with a team rather than individual development, a slightly laxer approach was adopted during the project. A toned-down version of XP (‘Extreme Programming’) was followed. The PID laid down a group of processes that should be followed, but soon after project start only the most effective ones were followed strictly. The XP processes followed were –

1. Keep a spreadsheet of required features, prioritized.
2. Define specific engineering tasks to get done (This ended up being done on paper as there were many small tasks could be dealt with quickly) A notepad was always kept at hand.
3. Time-box each session. (This was done for most (but not all) sessions). It did become clear that on average more work was done during time-boxed sessions, and it made it easier to quantify exactly how much had been done in a specific amount of time.
4. Utilise effective versioning (Using git as discussed).
5. Frequently reprioritize Gantt spreadsheet appropriately.

### Legal, social, ethical, and professional issues

Coding standards eg. DRY coding

### Stage 1

Look at highlight reports for stage 1

Planning – talk about designs that I drew, user stories

### Stage 2

Look at highlight reports for stage 2

### Stage 3

Look at highlight reports for stage 3

### Stage 4

Look at highlight reports for stage 4

### Project post-mortem

### Conclusions

## Statement of word count

## Reference List

## Bibliography

## Appendices

### User Guide

### Project Management Artefacts

### Other materials (UMLs, designs, test results)

### 

Analysis -One “definite” aim was that it should be a “puzzle-based game”. This did not end up being completely true, as only one of the games ended up being a puzzle based game (the ball-in-pipe game). The others games are a mix of action (archery), creativity (painting) and just a relaxing experience (snowy scene). The next aim was that the game should feature

1. https://en.wikipedia.org/wiki/The\_Lab\_(video\_game) [↑](#footnote-ref-1)
2. https://www.vrs.org.uk/virtual-reality/history.html [↑](#footnote-ref-2)
3. https://vr.google.com/cardboard/get-cardboard/ [↑](#footnote-ref-3)
4. <https://store.google.com/product/google_cardboard> <http://www.samsung.com/global/galaxy/gear-vr/> [↑](#footnote-ref-4)
5. http://www.techradar.com/news/wearables/htc-vive-vs-oculus-rift-1301375 [↑](#footnote-ref-5)