The Lab 2

Final Report for CS39440 Major Project

Author: Shankly Richard Cragg (shc27@aber.ac.uk)
Supervisor: Dr David Hunter (dah56@aber.ac.uk)

1st May 2018

Version 1.0 (Final)

This report is submitted as partial fulfilment of a Bsc degree in Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name Shankly Richard Cragg

Date 1st May 2018

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Shankly Richard Cragg

Date 1st May 2018

Acknowledgements

I'd like to thank my supervisor, David Hunter, for being a valuable source of knowledge, and great sounding board for ideas

Jon Shire, for guidance on writing fantastic reports

My Mother, Michelle Cragg, and Father, Philip Cragg, for everything

Abstract

With the ever-increasing presence and development of virtual reality in the modern world [1], exploring virtual reality as a medium is of paramount importance as mainstream appeal grows.

This project takes inspiration from Valves “The Lab” [2], and Owlchemy Labs “Job Simulator” [3], and looks to create an experience which is difficult or dangerous to recreate in physical reality, taking full advantage of the strengths inherent in virtual reality as a medium.

“The Lab 2” is therefore an introductory tool to the possible experiences virtual reality can give which cannot be taught better any other way. Specifically, this project looks to give the unusual experience of being a Steam Engine Fireman [4], who must keep his train running under the Drivers orders.

With a coherent play space, and audio/visual feedback, this experience is functional as a user’s first virtual reality experience, while offering something to those who are familiar with the platform.

This report details the process of a developing with virtual reality for the first time, and the process used to ensure development went as smoothly as possible. With analysis of the challenges a 1-man team faces when trying to develop a feature-rich and complete experience.

Contents

1. Background

1.1. Background Research

Valve Corporation [5] is an American video game developer and digital distribution company. They are the developers of a video game released in April 5th, 2016 called "The Lab". This was Valve attempting to "Understand existing genres though the lens of VR" [6]. The Lab features 8 short and varied experiences, ranging from Archery to a visual representation of the Solar System.

"The Lab" has the difficult task of being the first virtual reality experience for a huge number of consumers, not only in 2016 when the game was released, but even now in 2018 and beyond. Valve accomplish this with comical slapstick humour and simple intuitive controls and tutorials aplenty. The Lab is easy to pick up and play with no prior experience with the virtual realm to this day, with the game's polish and wide appeal a testament to the developers.

This project aims at acting as a spiritual successor to the fun fuelled mini-game collection by Valve, with a similar focus on a unique scenario and fun intuitive interaction with Objects and the Environment.

1.2. Ideation

The inspiration for the theme and feel of "The Lab 2" and the Steam Engine Fireman idea came from Owlchemy Labs "Job Simulator". This game features a range of real world jobs turned comical such as "Auto Mechanic" and "Gourmet Chef". These exaggerated experiences take place in an environment many users will be intimately familiar with. For the project a decision was made to take the dynamic of hectic interactive gameplay in a workplace and look for a dangerous or difficult to recreate scenario in the real world to apply these criteria to.

After some brainstorming, I came upon the role of a "stoker", or "fireman", whose job it is to tend to the fire of a steam engine, by shovelling coal into the engines firebox.

1.3. Motivation

My motivation for this project stems from the exciting range of experiences afforded to me during my Industrial Year. I had the chance to work with Augmented Reality (AR) using the Microsoft HoloLens [7], which similarly to virtual reality development can make use of cross-platform game engine Unity [8]. This experience with AR development was a joyous one and having been introduced to virtual reality for the first time with their internal HTC Vive kit gave me great curiosity into immersing myself into the world of virtual reality development. This project is a great opportunity to further develop this skillset.

2. Analysis

2.1. Existing Projects

2.1.1. The Lab

The project is to create a virtual reality experience, and as such my research involved the playtesting of existing games found on the Steam platform.
The research done showed that the best VR experiences focus on interaction as main motivation of play. For example, in figure 1 below, Valves “The Lab” allows users to travel to exotic and beautiful spaces to explore, including a Venice town square, or the top of a mountain in the Icelandic wilderness. However, the captivating thing to do in these areas is play with some sticks present in all locations. Throwing them, juggling them, watching them bounce off things. This focus on physical interaction and gameplay came as the standout experience even in the face of such beautiful vistas.

Figure 1 "Postcards" from The Lab

Source: Gameplay from “The Lab”

This ensured that the very core of my gameplay loop was going to encourage constant interaction with the elements in the world.

2.1.2. Job Simulator

Something Owlchemy Labs “Job Simulator” had to offer was the vibrant and expansive experience that can be offered in small spaces.

Figure 2 Job Simulator Office Level

Source: Steam page for Job Simulator

In Job Simulator offers a few diverse types of interaction. There are static objects, such as the keyboard which allows the user to press buttons or pull levers. There are also more free form objects such as the paper airplane, with its dynamic flight physics, and doughnuts, which the user can move into their mouth to eat. As well as coffee cups you can fill with beverages to drink and plugs which can be pulled out or plugged it.

The variety of interaction in such a small space was compelling gameplay. As a 1-man project, it made a huge amount of sense to limit the design space to as small a space as possible in order to reduce the amount of environment design necessary, an area I feel particularly uncomfortable with going into the project.

2.2. Hardware

Regarding hardware there are 2 main competitors when it comes to virtual reality Systems and Head Mounted Displays (HMDs). These are the HTV Vive, and the Oculus Rift [9]. These direct competitors have a lot in common, and the choice of platform for most users comes down to price point and experiences which are exclusive to each system.

For this project, I am developing with the HTC Vive, as the University Campus features a room complete with a HTC Vive system. Because of this, I will also acquire a HTC Vive to aid development off-site and keep playtesting as convenient and available as possible.

2.3. Game Engine

Similarly, there are 2 options concerning game engine to develop on as well. These are Epic Games “Unreal Engine 4” [10], and Unity Technologies own “Unity”. Both are cross-platform game engines free to use for development by individuals or for educational purposes. Once again this comes down to personal preference. I have some prior experiences working within Unity, and when researching development kits, the most popular one found named “VRTK” (Virtual Reality Toolkit) is exclusive to Unity.

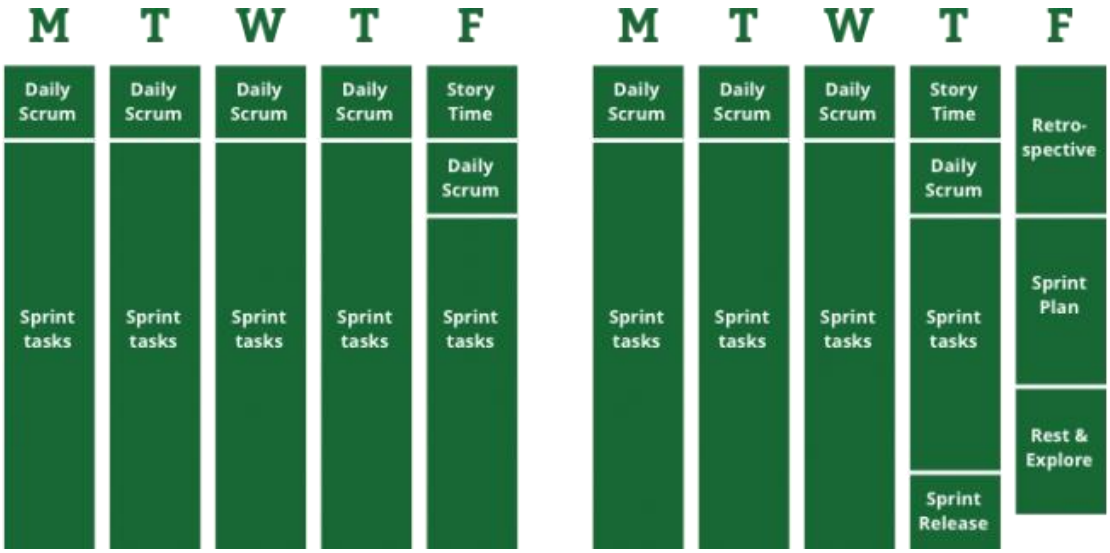
3. Process

3.1. Scrum for 1

3.1.1. Sprints

For this project, I decided to use a modified version of the agile methodology Scrum. When researching single team development practices, I came across an article from Alex Andrews titled “Scrum Of One: How to Bring Scrum into your One-Person Operation” [11]. Here he discussed a 2-week sprint cycle, shown in figure 3 to visualize his process.

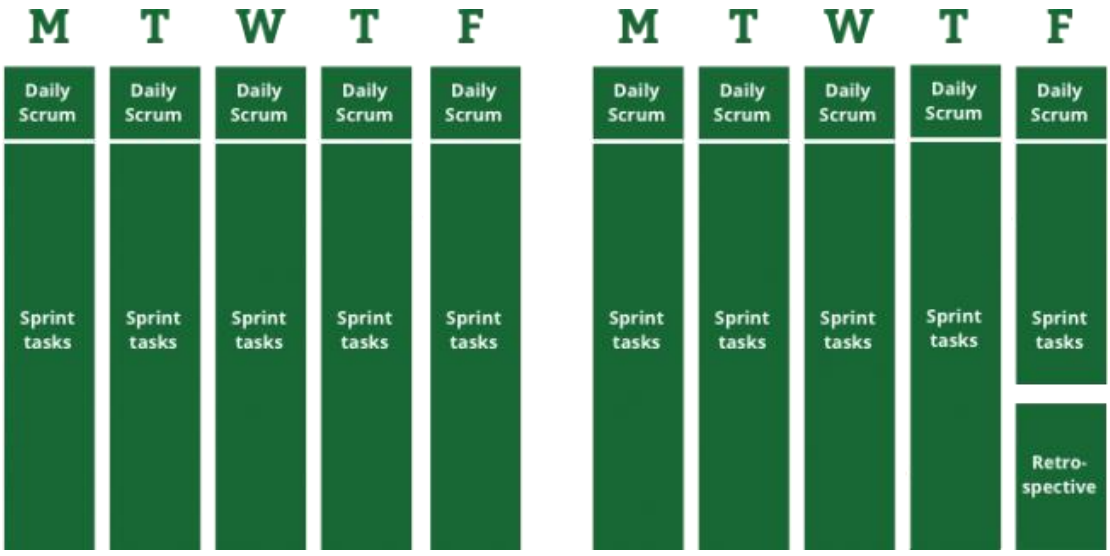
Figure 3 Alex Andrew's 2-week sprint



There were a few changes made to this schedule. The major made change to the schedule was the removal of “Story time” as a time dedicated to the direction of the project. As the project was working in unfamiliar space, sizings and the direction of the project was going to be incredibly fluid. This was instead done dynamically during the Daily Scrum at the start of each day, and reflectively at the end of the day as I wrote my daily diary.
The “Big picture” would instead be thought about in the bi-weekly retrospective. At the end of every 2 weeks the previous week would be reviewed, with concern regarding about which tasks were easier, and which tasks were harder than expected. This allows maximum focus on development time and gaining experience with the technology and give more leeway to on the fly decisions being made as new knowledge is gained.

This agile approach suits the project due to the open-ended nature of the project, and the lack of experience working with the technology. An agile process allows more flexibility over the day to day activities compared to a plan driven approach. The streamlined process used can be seen in figure 4.

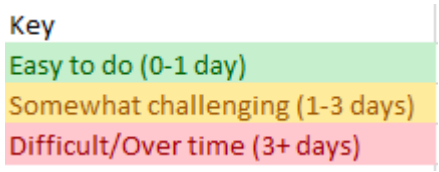
Figure 4 Adapted variation on the 2-week sprint



To keep track of tasks to be done, were in the process of doing, and had been completed, a task list was kept in an excel document. This had a key for the time it would take to accomplish a task seen in figure 5.

3.1.2. Task list

Figure 5 Key for the task list



These tasks were considered at the start and end of every working day as I wrote my daily diary. As iterations continued, new tasks could be added, or their expected sizing changed based on previous experiences. The main goal of this was to ensure that what amounts to a sprints worth of work was being completed every sprint. Naturally this wasn't always the case due to inaccurate sizings, but this made sure progress was consistent, and kept track of while multiple tasks were being developed.

At the start of the project, a few user stories were created to give some concrete end goals for the project. Some of these are epics which struggled to be narrowed down to smaller concrete user stories, such as;
"As a player, I want a high FPS so that I avoid feelings of motion sickness and lack of immersion."
This is not a goal which can be completed within a single sprint, and it cannot be compartmentalised into smaller stories, as optimisation of individual parts of the system can only come about after each in game element is added. The stories mentioned can be seen in Appendix 10.3.

3.2. Documentation

As part of using scrum, both a daily diary and updated task list were used to keep track of progress on various aspects of the project.

The code is documented with XML documentation for all classes and functions. XML documentation is generated at compile time, and can be distributed with the .NET assembly such that IDEs such as Visual Studio can show information about the classes and functions within.

The codes progress was also documented via frequent commits to GitHub, whose version tracking software keeps a record of all changes made to the program alongside my comments on what was done.

4. Restrictions of VR

4.1. Effective Resolution of Head-mounted Displays

We mention effective resolution when discussing the reality of using Head-mounted Displays, because looking purely at the number of pixels on each screen doesn't give enough information to properly convey the quality of the image.

When discussing HMD resolution, what matters is called "*pixels per degree*" or ppd. The reason this is a better metric than others such as total pixels, or pixels per inch, is that it takes into account magnification.

The Vive has two displays, one per eye. Each individual display has a resolution of 1080x1200 for a combined resolution of 2160x1200.

Put simply, despite the relatively good resolution compared to your average 1920x1080 desktop monitor, the 2160x1200 pixels of the Vive are spread over a much larger area due to how close they sit to your face. This very large field of view that only a nominally larger number of pixels must cover leads to individual pixels appearing larger to the viewer, and therefore when using the Vive, users may report it looking "pixelated".

What this means for the project is that small, intricate details are harder to see clearly in Virtual Reality. This includes effects such as small text being near impossible to read, and complicated/intricate objects and textures being hard to properly understand.

4.2. Maintaining suitable frame rate

A common complaint from VR users is what the virtual reality industry refers to as "simulation sickness". This results from participating in virtual reality on a machine which cannot run the software at a high enough frame rate to avoid jitter or stuttering. The frame rate needed to avoid this with consistency, and what has become the standard for virtual reality hardware developer, is 90 frames per second. The HTC Vive comes as standard with 90Hz displays.

The HTC Vive for its minimum requirements asks for at least an Nvidia GeForce GTX970 graphics card, which alone costs upwards of £300 at the time of writing.

The reason for such high-end components being required is that the HTC Vive houses 2 displays, one for each eye. This means anything needing to be rendered must be rendered twice from slightly different positions because of the distance between the eyes, which is what gives the feeling of perspective. This is much more costly resource wise than simply having 1 screen doubled twice. Both screens must maintain the same frame rate of 90fps ideally.

The project was developed on a Nvidia GTX980 graphics card, which is slightly better than the minimum requirements for the HTC Vive, and is listed under the complete list of recommended graphics cards HTC provide [12].

This makes a key area of the project performance optimisation. Keeping the scene size in Unity as small as possible, with as few tricky things to render or collisions to calculate as possible to ensure this standard of 90fps is met, avoiding simulation sickness.

4.3. Play space size

According to Valve themselves, 3 in 4 VR users have a play area large enough to support roomscale VR [13]. This leaves a sizeable number of users, about a quarter, whose play area does not support roomscale VR. These users have access “Standing Only” play areas, which is roughly 1 metre by 1 metre on average.

The play space the project was developed in is classified as a “Standing Only” play area. With such a percentage of players being in this situation, and the project ideally being available to everything, the project should at least be functional in a standing only play space.

5. Objectives

- Consistent frame rate greater than 90 frames per second.
- Dangerous or difficult to recreate scenario.
    - Steam Train fireman
- Consistent interaction enforced through game design.
- The program must run on the HTC Vive.

5.1. The Task

Development of a dangerous or difficult to re-create situation with a spin to the educational side, has led us to the player taking the role of a gamified steam engine fireman/stoker.

The most well-known role of this job is to shovel coal to fuel the engine. A natural cause of this is that temperatures will increase. This intuitively implies that the system will need to be cooled in some way. This implication is a logical step to the second task that will need to be completed, which is keeping the temperature to a minimum.

Interweaving these gameplay elements, with a score system in place encouraging interaction while running at a high and smooth frame-rate are the goals to be completed.

6. Designing the Game

With the base elements of shovelling coal, and pouring water decided on, finding ways to make these elements interactive as possible, with an intuitive method of feedback needed to be found.

6.1. Fueling the Engine

There are 3 components to think about with regards to this activity. These are;

- Spawning of coal
- Interaction of moving coal into furnace
- Deletion of coal

Each of these components need to make sense in universe, progress the goal of interactive gameplay, and feel fun.

6.1.1. Spawning of coal

With the spawning of individual coal rocks, there are few ways to make this inherently interesting to a player. This is a means to an end to allow the player to interact with the objects being spawned.

Something inherently enjoyable in the VR space is motion, due to physically having to move your eyes and head to follow objects. The way we will spawn coal to take advantage of this will be from a pipe placed on the ceiling of the room, meaning all coal spawned falls and crashes to the floor.

The spawning will start slowly. This allows the player to see where coal is coming from, and have time to mentally take note of this. However, as time progresses, coal will spawn faster and faster to a maximum amount, increasing the chaos and intensity in gameplay.

6.1.2. Moving coal into furnace

The furnace the coal is to be moved into should be raised slightly so there is some effort required from the user, but the task isn't too finicky or tricky. The shovel should be good relative size with the coal to allow finesse from the user, but not make interaction harder than necessary.

6.1.3. Deletion of Coal

Once the coal is in the furnace, it needs to disappear somehow. A couple of methods are possible for this. One method is to despawn the coal after x seconds inside the furnace, however this is very interactive. Another option is to have the coal stay in the furnace but lose its physics properties and colliders so the user can see it build up in a satisfactory way. But this eventually would make it hard to see new coal entering the furnace, and has the same problems as the first with regards to lack of interactivity.

The method used takes some creative liberty with the real-world function of coal automatically being used, but adds that interactive element while still making sense in play. This is to add a crusher mechanism to the furnace, which the user must pull down to crush and destroy the goal, at this time the coal is then added to the system and destroyed.

This means coal is added in bulk now, which is a bit tricky for the user to control when they wish to add a certain amount. This can be seen as a skill within the game gained

through play however, where the user eventually intuitively figures out how much coal is needed to add X amount to the coal counter before pulling the crusher, and potentially over-fuelling the engine.

6.2. Cooling the engine

Like shoveling coal, this gameplay element has 3 components;

- Spawning of water
- Interaction of moving water to receptor
- Deletion of water

6.2.1. Spawning of water

Spawning the water will play off the design decisions of the coal spawner, and come from a pipe ejecting in the ceiling. In order to spawn the water, the user must pump a lever in order to emulate a real-world water pump, giving this task that necessary element of interactivity.

6.2.2. Moving water into receptor

To move the water there will be a bucket, which the user must place under the water spawner to fill up, before moving and pouring the water into the receptacle.

6.2.3. Deletion of water

The water will disappear into the receptacle quickly. There are no easy or sensible ways to force interaction into this part which came up in brainstorming, and with the need to spawn, collect and pour the water already existing, deletion of the water being as simple is not a concern.

6.3. Environment and motion

Being on a moving train, it is important to give the player the feeling of motion and speed.

Since the user is situated inside the train, the train becomes his frame of reference.

Importantly, all that matters are that the user feels like they are in motion, not that they are actually moving. This means there are 2 possible options for giving this experience.

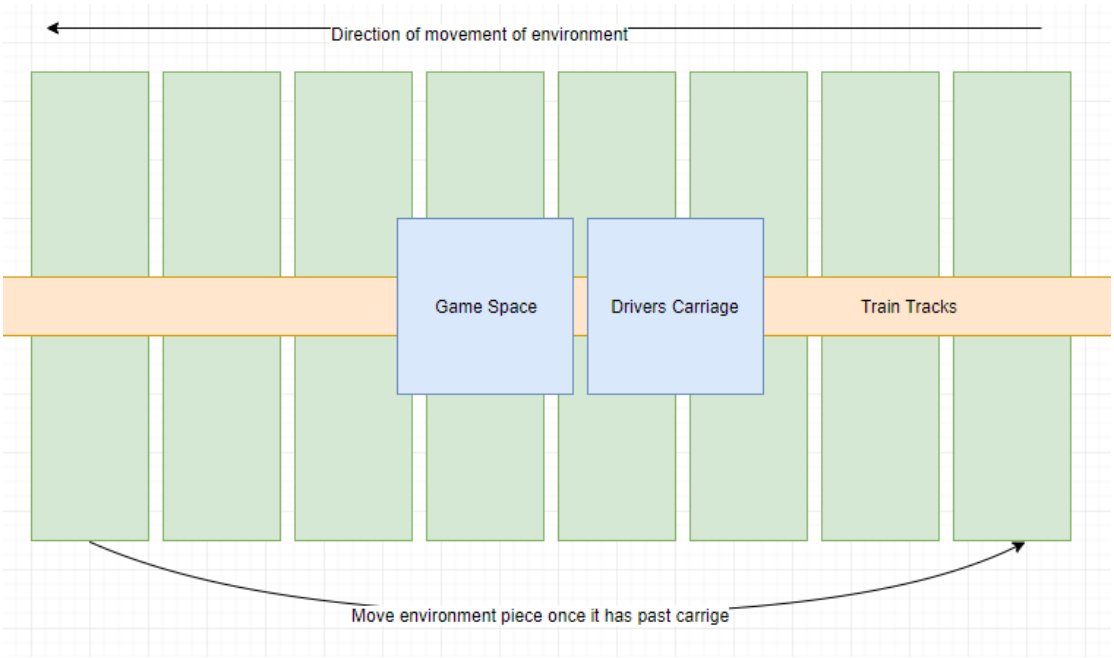
6.3.1. Static environment

The first of these options is to have a static environment. This would entail having the train itself gain and lose speed as it moves through the x or z coordinate space.

6.3.2. Static train

The second option, and the one chosen for the project, was to have the train remain stationary at the centre of the scene (Vector 0,0,0). Instead we have the environment move backwards past the train, to give the illusion of movement forward. This avoid any possibility of the tiniest jitters when changing speeds and such having unwanted effects on the gameplay area.

Figure 6 Environment design



Using this method, it is possible to reuse the same X number of environment prefabs, cycling them from back to front as needed. Limiting the number of environment pieces in the game to lessen the impact on performance.

6.4. Visual Feedback – Gauges

In order to give the player feedback, 2 gauges will act as the in-game feedback mechanism for the status of fuel and temperature.

Gauges are a useful mechanism due to their ability to scale to any size needed, be that small or large to best fit the aesthetics of the game space. They are also very good for getting feedback quickly, requiring only a cursory glance at to understand what they are saying. This is especially true as the user gains familiarity with the gauges.

The gauges will use a colour scheme like real world traffic lights, with green meaning good, yellow meaning warning, and red meaning bad.

These gauges will be the main way the player gathers information to decide which move to make next in gameplay.

6.5. Driver

As a mechanism to give feedback to the user about their performance, a driver in charge of the train, situated in the carriage in front of the players carriage to give live feedback and add a humorous human element to the project.

Feedback would be dynamically given based on how well the user was doing, with triggers points at certain thresholds of performance for dialogue.

6.6. Scoring

The goal of the game is to last for as long as possible. The user will start off in a moving train, with everything above board.

There are 2 metrics for scoring the player. A “score”, which is used to calculate how well the user is doing, and a “speed” of the train, which is determined by the performance of the user during gameplay. If the user performs poorly, the speed will drop, if the user does well, the speed will climb to a maximum.

Once the “speed” drops below a threshold of half the maximum speed, then the score will start depleting. The user can stop the score depleting by making both fuel and temperature back to optimal, so that the train gains back speed again.

If the user fails to maintain optimal fuel and temperature, and the score drops to zero, the game will end.

If the user fails to keep the speed of the train above zero, the game will end.

The motivation for the user is to last for as a long as possible. There is a timer in the game space, allowing the user to check how long the session is lasting. Once the game ends, this timer stops and the user can see how successful they were.

7. Implementation

7.1. Gauges

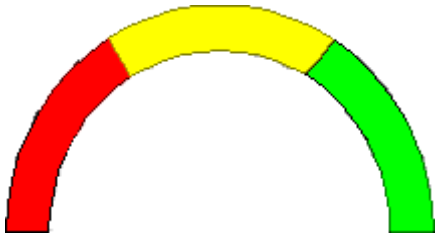
The first thing to be figured out was how to visually represent the fuel and temperature. The decision was to use gauges in the design stage due to their ability to scale well, and give feedback to the user quickly. First thing to figure out was the design of the gauge itself.

7.1.1. Design of Gauge

The first thing to be figured out was the design of the gauge. It had been decided to use traffic light colours of green yellow and red for good, warning and bad respectively. The temperature gauge is the simplest of the 2. With a large semi-circle to mitigate the somewhat low effective resolution of current VR technology, and 3 strikingly separate sections to easily see at a glance. Pictured below in figure 7 is the temperature gauge used in game.

The ratios for temperature are kept deliberately simple, with each third of the gauge representing one of the three states.

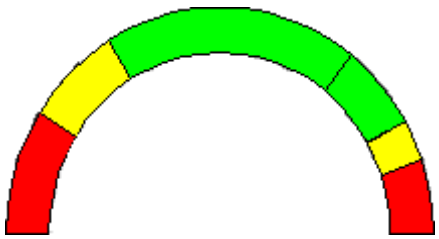
Figure 7 Temperature Gauge



For fuel however, the decision was made for too little fuel to be a problem, but also too much fuel. Balancing out the ratios for this was going to be an important decision as to how badly the player should be punished depending on whether they are putting in too much or not enough fuel.

It makes a lot of common sense to punish the player more harshly for a lack of fuel, as opposed to putting in too much. This is the especially the case due to a design decision made earlier in the project regarding the crusher. If fuel is entering the system in bulk as the user pulls the crusher, then it will be easy to overshoot the goal and input too much fuel. So, to avoid punishing the player too much, the yellow warning and red danger zones are half as large on the “too much fuel” right hand side of the gauges, as the “too little fuel” zones on the left-hand side. Below in figure 8 you can see how this weights the gauge to one side slightly.

Figure 8 Fuel gauge



7.1.2. Rotation in Unity

The gauges have a central needle which acts as the pointer for each gauge. Manipulating this needle via rotation in Unity gave way to some interesting emergent design.

Within Unity, rotational are represented 2 different ways. Quaternions and Euler angles. Quaternions are used internally by Unity, but within the editor itself, Euler angles are used as they are more intuitive to how humans normally understand rotation.

Euler angles are a human readable format for representing rotation, which simply has three angles (x, y, z), all applied sequentially. For example, a Euler angle of (0,0,0) is no rotation, and (0,180,0) is a 180 degree turn around the y axis.

Quaternions on the other hand take 4 values (x, y, z, and w). These do not represent the axis and their angles in the way we normally associate x, y and z coordinates. The math behind quaternions is quite complex, and official Unity documentation for quaternions and Euler angles in Unity recommends against going into any depth regarding quaternion math unless "particularly interested" in directly modifying quaternion values specifically [14], which we do not need to do for our project.

Quaternions are used internally because they avoid a common problem in dealing with 3D rotation with plagues Euler angles known as Gimbal Lock. With three degrees, if two of the dimensions become parallel, then the system becomes locked into a 2-dimensional space for rotation. Once parallel, changing either of the locked-up dimensions also changes the other in the exact same way.

The nice thing about Unity is that you do not need to understand the underlying theory behind quaternions as how they operate to code in, and make use of rotation.

7.1.3. Relation between coal and temp

Naturally, the elements of fuel and temperature decrease over time. The rate at which fuel depletes increases as the game progresses to a maximum, however the rate of temperature decline stays the same. With the fuel now having states of "too much", "just right" and "too little", we can add in adjustable temperature increase too, with the increase in temperature corresponding to the state of fuel.

Fuel State	Temperature increase
Too much	Increase fast
Just right	Increase at base amount
Too little	Increase slowly

This has the benefit of helping the player continue in tricky scenarios and encouraging action when fuel is too high.

If there is too much fuel, it would take a while for it to eventually reach a state where there is too little. By increasing the amount, the user has to tend to cooling the temperature at this time, we give the player a similar amount of urgency as when both mechanics need to be tended to. It also allows the player some breathing room when fuel is too low to fix that issue, since the slowing of temperature increase means the user can focus on the coal task for an extended period of time.

This yin and yang of the 2 game elements means the user can never rest on their laurels during a time when both tasks are at optimal amounts, but also that they are given some leeway when they get into more tricky scenarios.

7.2. Coal Spawning

7.2.1. Randomised spawn times

To give the game a sense of progress and raising chaos, the coal gauge depletes quicker and quicker (to a maximum) as the game continues. To match this, the coal spawner starts of spawning coal slowly with a large interval between pieces. Over time, the interval between spawns shrinks to a minimum and through this the world matches the players frantic nature as the speed of tasks to be completed increases.

7.3. Water Spawning

7.3.1. Shower Effect

To make the water spawning have a shower / spray like effect, it uses a modified version of the logic used for the coal spawner, as both of these classes inherit from a "spawner" class. The water implementation however sets the interval for each spawn extremely low, and randomises the position of each spawn within the spawn entity, in this case a pipe in the ceiling. This has the effect of every single particles having a slightly different height as it falls, and slightly different position. It replicates the shower effect very well and feels intuitively like a spray of water.

7.3.2. Coroutine for pump

With the water pump, 2 states exist. Unpumped when pulled to the top, and "Pumped" when pulled down to the bottom. For the water to spawn, the state must change from unpumped to pumped. This ensures that after the user pulls down the lever, they must have pulled the lever all the way up before pulling it back down again.

To ensure a reasonable but fair amount of water spawns at each pump, the spawner is turned on for some seconds using a coroutine. This prevents bugs like the user just holding the pump at the down position to infinitely spawn water, or each pump only spawning a tiny amount of water. The coroutine allows a controlled amount of water to spawn each time, as activating the spawner for some seconds allows consistency in the chaotic action of pumping this lever.

The coroutine simply yields for some seconds, before disabling the spawning script in the rest of the function. Yield allows the developer to introduce a time delay before continuing the function. Thereby enforcing the spawning stays active for the right amount of time.

7.3.3. Inverse Meshes with bucket, instead make custom bucket

The shovel in the game uses a free asset found on the Unity Asset store. Originally there was also a free bucket asset found on the Unity Asset store as well, however using it as a bucket had its difficulties.

Unity 5.0 stopped supporting non-convex mesh colliders, meaning that physics interactions with shapes such as a bucket, with an internal area cannot exist anymore. Essentially turning your bucket into a cube, as the top acts as if it is closed off. The solution to this is to make the bucket out of small convex mesh colliders placed together. This fix works for all shapes and sizes, usually by re-creating the original wanted shape with many small cubes, building the shape out of Lego essentially. The smaller the cubes and larger amount you use, the more accurately you can recreate the original object up to near perfection.

This is the solution used in the project. Accomplished as simply to possible, the bucket has 4 sides and a bottom to it. The smallest number of cubes necessary to create the original function. This does not come with the pretty textures an asset from the unity store would come with, but it does shrink the project size and mean this part of the project is now entirely original.

7.4. Aesthetics/Audio

The most unpolished aspect of the project is the visuals and audio. The project suffered from lack of experience and a different skillset required to create a beautiful, realistic looking scenario, with fitting and subtle audio. Any visuals in the project which look as good as everything should look were taken from the Unity Asset store, as free assets. Audio was harder to find, and while experimentation was done into using audio with the coal spawner (which can still be used by enabling the audio files attached to the coal spawner in the Unity editor). However, this audio was jarring and having it add to the experience was something not accomplished during the time frame.

7.5. Textual feedback

7.5.1. Timer

The goal is to last for as long as possible before failing. To keep track of the time, a 3D text object is placed on the back wall of the game space. This area is unused, so having an item the player only needs to look at out of curiosity gives a reason to take a glance behind them.

The timer text is large, and this is to combat the discussed low effective resolution of current HMDs. The large text may still appear blurry at the edges, but it will always remain readable at the scale it exists, which is the most important goal.

7.5.2. Score and Speed

The current solution to conveying this information is functional, but not ideal. The speed of the train is naturally conveyed somewhat by the actual speed the player feels as they travel past the environment, but due to the scoring system happening at an exact threshold of half the maximum speed, it feels only fair that the user has direct reference to the current status.

The way the user can see their score and speed is similar to the time, except this is placed in the driver's carriage, which makes sense from a design perspective, as the driver is supposed to act as the person in charge who would be responsible for knowing these stats.

In a world where the driver is more fleshed out, he would be able to give the user the information needed verbally or through his physical appearance changing and animations. However, these aspects proved too time consuming and difficult to learn in the time-span of the project, so a functional solution was focused on.

7.6. Optimization

7.6.1. Preventing coal build-up

When looking to optimize the performance of the project once everything was implemented, the first place to look was to stop objects infinitely spawning which would

eventually lead to a very large amount of physics simulations that need to be run every frame, and objects to be rendered, all having a massive effect on performance.

When brainstorming how to keep the amount of coal in the game space from increasing forever, focus was on finding a solution that improved the experience. The best way to do this is to add another method of interaction requiring attention from the user.

What came from this was the idea to have the coal spawner “break”, requiring fixing by the user to continue spawning coal. What happens is every time a piece of coal is spawned; a function is run to decide if the spawner should break or not. The odds of this can be changed, but it sits at 50/1000, or 1/20 currently.

Once the spawner breaks, an audio queue plays (disabled in built version due to garish noise), and smoke bellows from the spawning pipe.

To fix the spawner, the user must “hit” the spawning pipe 3 times, at which point the smoke disappears, and coal will continue to spawn.

In order to achieve this, we hooked into VRTK to create the touch event with the HTC Vive controllers and the spawning pipe. The VRTK library comes with many events for touching, grabbing, gripping and more. Creating a custom function to count how many times the user has interacted with the spawning pipe was made much easier due to the existing touch events already existing within the VRTK library.

With this in place, the game now stops spawning coal indefinitely, maintaining a high frame rate even is the user negates crushing existing coal. The only way for coal to have an impact now is if the user purposely does not crush coal to remove existing pieces, and fixes the spawner every time it breaks. This is not really a concern however, as if the user spends this long not crushing coal, the game will have ended.

7.6.2. Reducing water particle effect on performance

Large water particles for reduced performance + ease to run, and fun to play with
Water despawn after 13 seconds to maintain low number

We needed to do a similar optimisation to the coal pieces with water particles to ensure they are not causing copious amounts of stress to the system. With the pumping system in place to spawn water, players who are ignorant of the water system are no danger to indefinite water spawning.

The system spawns water objects, where each individual droplet is its own object with physics properties. This is for gameplay reasons, for the user to catch the “water” in their bucket, and then pour it into the receptacle.

To optimize this aspect there are some options. Firstly, we can make the droplets as simple as possible, so that each one causes as little strain as possible. The mesh render has some options to reduce performance impact, with the largest offender being that the droplets can cast shadows. Shadows are a relatively expensive process to calculate and render, and so removing the near unnoticeable shadow from the water helps improve performance. People often see water as transparent, so water not leaving a shadow does not come across as startling or out of the ordinary.

Secondly, we can optimise the number of droplets needed. This is accomplished by increasing the size of each droplet, and reducing the amount spawned, such that the same amount of space is taken up by the water in fewer droplets. Finding the balance

between large enough to not cause performance issues, and small enough to reasonably resemble water droplets is up to interpretation. Most importantly, spawning water must still feel like a shower, which is a feeling kept despite increase in size and decrease amount.

Lastly, we want to prevent the possibility of a large amount of water droplets building up in the world. Due to the nature of the water spawner, a large number of particles can be made every second, and as such this issue needs to be tackled aggressively. The solution to this is to add a lifespan to each droplet, currently set to 20 seconds. This gives the user ample time to fill their bucket and pour it into the receptacle. This also has the added benefit of users no longer being able to make the game easier by pre-filling a bucket of water to be used next time they need it, enforcing gameplay.

7.6.3. Reducing environmental effect on performance

Remove colliders and rigid body physics meshes from all objects not internal to train

The environment is composed of trees from the Unity standard assets to make up the forest, and a simple cube stretched to make the ground.

Due to motion given from the environment moving past the static train car, the environment is moving a lot, and this had a humongous performance at first. This was so detrimental to performance it made the game unplayable once implemented, resulting in frame rates approaching the lower single digits.

The reasons for this turned out to be that each tree from the standard asset set was made up of several collision and physics components, so that they could be interacted with and physically exist. However, for the projects implementation, the environment was never going to be explored or interacted with. All that was needed is the visuals. Editing the forest prefab, a set of objects used as a strip of forest which is used many times over to create the entire forest was a breeze in Unity. Removing all physics and collider components from objects is made simple thanks to the component design of Unity.

Once these were removed, the environment now had near zero impact on performance other than rendering it, which thanks to the optimizations inherent in Unity's standard assets with dynamic rendering based on distance, leaves almost no performance impact at all.

8. Testing

VR experiences, and Unity games in general are incredibly dynamic in their nature. This project does not feature input boxes which require string validation, or a server-side login system with security concerns. Instead the tests will take a black-box approach. This ensures the system functions for the user.

Test Case ID	Test Step	Expected Result	Observed Result	Pass/ Fail
1	Double click .exe	The game loads	The game loads	Pass
2	Turn on Vive controls, look at them with HMD	The controllers are visible in the game world	The controllers are visible in the game world	Pass
3	Press right grip button on right controller	Shovel appears in front of controller.	Shovel appears in front of controller.	Pass
4	Place controller inside shovel	Shovel is highlighted	Shovel is highlighted	Pass
5	Press trigger while controller is inside shovel	Shovel is grabbed	Shovel is grabbed	Pass
6	Release trigger while grabbing shovel	Shovel is dropped	Shovel is dropped	Pass
7	While holding shovel, have shovel interact with coal	Coal is moved	Coal is moved	Pass
8	Place controller inside bucket	Bucket is highlighted	Bucket is highlighted	Pass
9	Press trigger while controller is inside bucket	Bucket is grabbed	Bucket is grabbed	Pass
10	Release trigger while grabbing bucket	Bucket is dropped	Bucket is dropped	Pass
11	Throw bucket out from opening	Bucket does not leave train	Bucket does not leave train	Pass
12	Begin the game, look at the timer	The timer is at or near zero, and counting up	The timer is at or near zero, and counting up	Pass
13	After ending the game and reloading the level, look at the timer	The timer is at or near zero, and counting up	The timer is at or near zero, and counting up	Pass
14	After ending the game and reloading the level, look at the score	The score is at the original amount	The score is at the original amount	Pass

15	After ending the game and reloading the level, look at the speed	The speed is at the original amount	The speed is at the original amount	Pass
16	After ending the game, press the trigger	The game is restarted	The game is restarted	Pass
17	The score reaches zero	The game ends and the timer stops	The game ends and the timer stops	Pass
18	The speed reaches zero	The game ends and the timer stops	The game ends and the timer stops	Pass
19	With the water lever at its top most position, pull it down	Water spawns	Water spawns	Pass
20	With the water lever at its lowest position, move it up halfway, then pull it back down to its lowest position	Water does not spawn	Water does not spawn	Pass
21	Place controller inside water pump lever	Water pump lever is highlighted	Water pump lever is highlighted	Pass
22	Press trigger while controller is inside water pump lever	Water pump lever is grabbed	Water pump lever is grabbed	Pass
23	Release trigger while grabbing water pump lever	Water pump lever is dropped	Water pump lever is dropped	Pass
24	While holding water pump lever, try to move from wall	Water lever stays attached to wall	Water lever stays attached to wall	Pass
25	Both gauges are in the green, speed is 20	Speed stays at 20	Speed stays at 20	Pass
26	Both gauges are in the green, speed is below 20 but above 10	Speed increases	Speed increases	Pass
27	Both gauges are in the green, speed is below 10	Speed increases, score stops depleting	Speed increases, score stops depleting	Pass

28	Fuel is yellow, temperature is green	Speed slowly decreases	Speed slowly decreases	Pass
29	Fuel is red, temperature is green	Speed decreases	Speed decreases	Pass
30	Temperature is yellow, fuel is green	Speed slowly decreases	Speed slowly decreases	Pass
31	Temperature is red, fuel is green	Speed decreases	Speed decreases	Pass
32	Fuel is yellow, temperature is yellow	Speed decreases	Speed decreases	Pass
33	Fuel is red, temperature is yellow	Speed decreases quickly	Speed decreases quickly	Pass
34	Temperature is yellow, fuel is red	Speed decreases quickly	Speed decreases quickly	Pass
35	Temperature is red, fuel is red	Speed decreases at maximum rate	Speed decreases at maximum rate	Pass
36	Speed is below 10, both gauges are not green	Score decreases, speed decreases	Score decreases, speed decreases	Pass
37	Speed is above 10, both gauges are not green	Score stays the same, speed decreases based on gauge positions	Score stays the same, speed decreases based on gauge positions	Pass
38	Pump water, wait 20 seconds	Water droplets despawn after 20 seconds	Water droplets despawn after 20 seconds	Pass
39	Place bucket under water spawner, pump water	Bucket contains water droplets	Bucket contains water droplets	Pass
40	Deposit water from bucket into water receptacle	Water despawns, temperature gauge moves towards green	Water despawns, temperature gauge moves towards green	Pass
41	Start game, look at coal spawner	Coal spawns after a random amount of time	Coal spawns after a random amount of time	Pass
42	Start game, look at coal spawner and wait	Coal spawner breaks after random amount of time, causing coal to stop spawning, and smoke to bellow from coal spawner	Coal spawner breaks after random amount of time, causing coal to stop spawning, and smoke to bellow from coal spawner	Pass
43	After coal spawner has broken, “touch” the spawner	The coal spawner is fixed. Smoke no longer bellows from spawner,	The coal spawner is fixed. Smoke no longer bellows from	Pass

	with a Vive controller 3 times	and coal begins to spawn again	spawner, and coal begins to spawn again	
44	Start game, and watch coal spawn from coal spawner	Coal spawns faster on average as game progresses	Coal spawns faster on average as game progresses	Pass
45	Shovel a single piece of coal into the furnace, and pull down the crusher	The piece of coal is destroyed, and the fuel gauge is increased slightly	The piece of coal is destroyed, and the fuel gauge is increased slightly	Pass
46	Shovel 5 pieces of coal into the furnace, and pull down the crusher	All coal pieces are destroyed, fuel gauge increases by 5 times the amount of 1 coal crushed	All coal pieces are destroyed, fuel gauge increases by 5 times the amount of 1 coal crushed	Pass
47	Speed is 20, look out of train at environment	Large sense of speed in forwards direction, trees move past train from front to back	Large sense of speed in forwards direction, trees move past train from front to back	Pass
48	Speed is 10, look out of train at environment	Moderate sense of speed in forwards direction, trees move past train from front to back	Moderate sense of speed in forwards direction, trees move past train from front to back	Pass
49	Speed is at 0 (game is over), look out of train at environment	No sense of speed of motion. Trees are not moving	No sense of speed of motion. Trees are not moving	Pass
50	Score is at 0, speed is greater than 0	Speed and score are no longer changing. Train stays in motion	Speed text does not change, train continues to slow to a stop	Fail
51	Speed is 0, look at timer	Timer is stopped	Timer is stopped	Pass
52	Score is 0, look at timer	Timer is stopped	Timer is stopped	Pass

9. Critical Evaluation

9.1. Were the Objectives correctly identified?

I found writing requirements for this project an incredibly difficult part of the process. My previous work prototyping with augmented reality in Unity during my Industrial Year was a nice introduction to the use of Unity, but not a lot more when it came to create a work such as this as a 1-man unit.

This left the requirements somewhat vague as after some experimentation, I was not sure exactly what the process or expected outcome of the project would be until I was already many weeks into development and had a better sense of the scale and size of individual tasks.

The requirements I did set however were a good outline for a higher-level view into what needed to be done, and kept an end goal in mind.

9.2. How correct were the design decisions?

The design decisions turned out very well, with the only exception being that of the driver who was not implemented as I would have liked, but functional solutions to the problems he would have addressed were found.

Overall the game feels fun to play and encourages interactivity exactly as I had hoped it would, and I think this is in big part thanks to the design of the game making it easy to create such emergent gameplay, and balance it to a point where it seems fair and balanced. Each implemented design feature feels fleshed out barring visuals, and the interactivity between components was made a breeze thanks to designing the interactions beforehand.

9.3. Were the Objectives identified in section 5 met?

Ensuring the project stayed at over 90 frames per second was a huge goal for myself during development just so that I could play the game and see how things were going, so meeting this requirement was as much for myself and my own enjoyment, as for the project. I'm very satisfied in saying on most, if not all VR ready systems, the project will maintain a sustainable 90 frames per second or better.

The situation by its nature is one which is difficult to recreate, and potentially dangerous. The choice of scenario met this requirement very well.

The game run on the HTC Vive with no issues, making use of both the head-mounted display and the controls.

Perhaps most importantly, the player is constantly pushed to play the game, and the few mechanics are made excellent use of to encourage consistent gameplay and interaction. The dynamic interaction between the fuel and temperature needs especially help to keep the player having something to accomplish.

9.4. Could a more suitable set of tools have been chosen?

As mentioned in the Analysis section, there were other options for both hardware and software. Having now worked with these for a number of months, I have to give praise to Unity as a platform for development in particular. The engine makes many tasks that could be extremely tricky such as physics and rendering which need to be in place

before you even begin creating a product. Building an executable file to run is made as easy as a single digit number of clicks and works perfectly. Its user interface is customizable and intuitive once you learn the way everything meshes together. While I have no doubt Unreal Engine 4 could have also been a good choice, I don't believe it would have been substantially better, or even equal to that of Unity.

Between the start and end dates of the project, I got a chance to try the Oculus Rift, and while I was impressed with the ease of setup when compared to the HTC Vive, the use of the kit itself was very similar, with nothing apparent in my short playtime separating the experiences of the Vive and Rift besides the controllers used. The project would have handled near identically no matter the hardware choice, and so no more suitable choice could have been chosen.

9.5. How well did the software meet the needs of those who were expecting to use it?

Upon demonstrating the project during the mid-project demonstration, feedback was very positive as to the fun and interactivity elements. I greatly enjoy playtesting the game as it reached the conclusion of the project, and feel it gives the experience in a manner that many can find value in. When I compare it to the inspirations of Valves "The Lab" and Owlchemy Labs "Job Simulator" I think it would fit into those packages well with some refinement and polish to the rough edges of its aesthetic design.

9.6. How well were any other project aims achieved?

Coding in C# for the first time, I was worried there might be a learning curve to using the language, but very quickly I felt extremely comfortable writing in C#. It felt like I had been coding in it for a while. This innate familiarity due to many similarities with Java meant that code quality, specifically with regards to following Microsoft's C# standards was no extra effort. The code even featured comprehensive XML documentation on top of this in an attempt to document as thoroughly as possible.

9.7. What wasn't achieved

The visual and audio side of the project is a mix on unfinished and untouched. While I'm very pleased with the environment design and sense of motion it gives, and the gauges are a neat intuitive feedback tool, there is more that needs to be done. There is some audio attacked to the coal spawner, but this is very unrefined, and it disabled in the final release. It can be turned on it unity by enabling the audio files attached to the objects as components, but for now they remain disabled due to detracting from the experience as the noises are garish and distracting. Also, the internal design of the train is functional but clearly not complete, with prototyping textures covering the ins and outs of the main play area.

9.8. What would change if starting again?

Brainstorming a project which required less on the aesthetic side of game design, and more on an interesting coding challenge would play to my strengths more. While I enjoyed seeing that side of development, with the short span of time we have in the grand scheme of things, the experience is missing a lot of polish, and even glaring visual elements of the steam train such as the actual room you play in are completely unfurnished and still in an alpha phase of development.

A more code focused project would avoid this issue, while allowing more exploration into what's possible in Unity, with the extra time not spent on environment / aesthetic design, a save/load system would have been implemented, or local storage of high scores.

10. Appendices

10.1. Third-Party Code and Libraries

10.1.1. Unity [15]

Developer: Unity Technologies
Version: 2017.3.1f1 Personal(64bit)

Unity Personal is available for use for free to students in education. The following quote is taken from Unity's education page [16].

"Students who are using Unity outside the classroom to build or refine their skills are welcome to download the free and fully-featured Unity Personal."

Unity Personal also has a financial threshold after which use of Personal is no longer permitted. This would require an excess of $100,000 in the last 12 months [17].

10.1.2. VRTK - Virtual Reality Toolkit [18]

Developer: TheStoneFox

VRTK is self-described as "A productive VR Toolkit for rapidly building VR solutions in Unity3d". It includes a lot of basic models and interactions which are necessary to even begin developing an VR experience. Including but not limited to:

- Locomotion within virtual space
- Interaction
  - touching
  - grabbing
- Body physics
- 3D models for the HMD and controls

This allows us to focus more on creating an immersive experience, rather than being bogged down in the complexities of modelling interaction between the real-world controls and the digital one.

In this project, 2 gameobjects from VRTK example scenes are used to detect a HTC Vive is in use and load the correct associated prefabs and models with help from SteamVR. SteamVR is required to be installed for VRTK to work with the HTC Vive.

There are 2 sources for VRTK, the Unity Asset store, and the VRTK GitHub repository. The developer StoneFox, when asked which version to use, mentions in the FAQ for VRTK the following statement.

"The GitHub master version of VRTK is always the most up to date version with more features and bug fixes, however it is not as stable as the GitHub releases or the Unity Asset Store version. It is recommended that to keep up to date with the latest features, the GitHub master version is used." [19].

Due to the expert opinion of the developer, we decided to use the latest GitHub branch of VRTK available at the time.

VRTK is released under the MIT license [20].

The source code for VRTK was unchanged for this project. There is a piece of code which hooks into the VRTK event list to listen for Vive controller touches. Specifically, the "FixCoalSpawner" script for a custom listener event. Note this did not require directly modifying VRTK code.

10.1.3. SteamVR [21]

Developer: Valve Corporation

Unity does not natively support HTC Vive use. A plugin called SteamVR when imported into Unity will automatically configure Unity environment configurations specifically for HTC Vive development.

SteamVR is released under the BSD 3-Clause “New” or “Revised” License [22].

This plugin was used completed unedited. All use of SteamVR is done via VRTK.

10.2. Ethics Submission

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
shc27@aber.ac.uk

Full Name
Shankly Richard Cragg

Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
The Lab 2

Proposed Start Date
29/01/2018

Proposed Completion Date
04/05/2018

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
My project is to create an educational tool as an introduction to Virtual Reality, and is intended to be a spiritual successor to game publisher Valves first VR title, "The Lab". Hence the name "The Lab 2". This is not a research based project, and as such does not require animal or human participation to gather feedback or gain knowledge.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?

Not applicable

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
No

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:

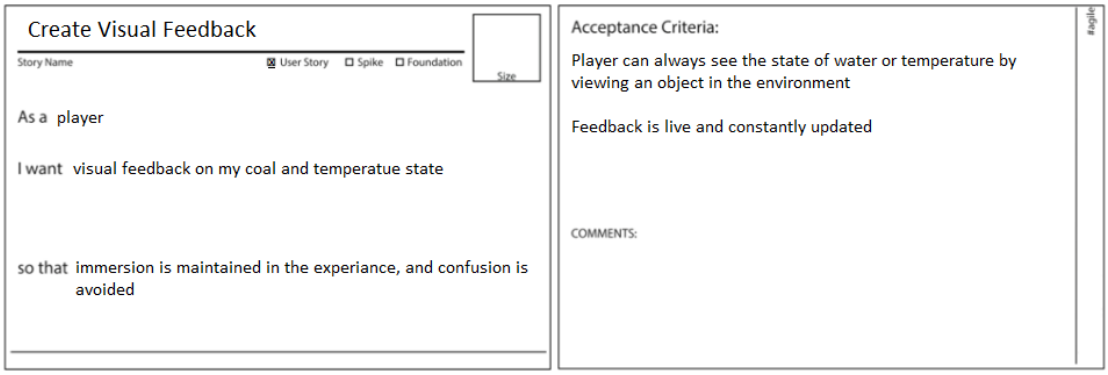
If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

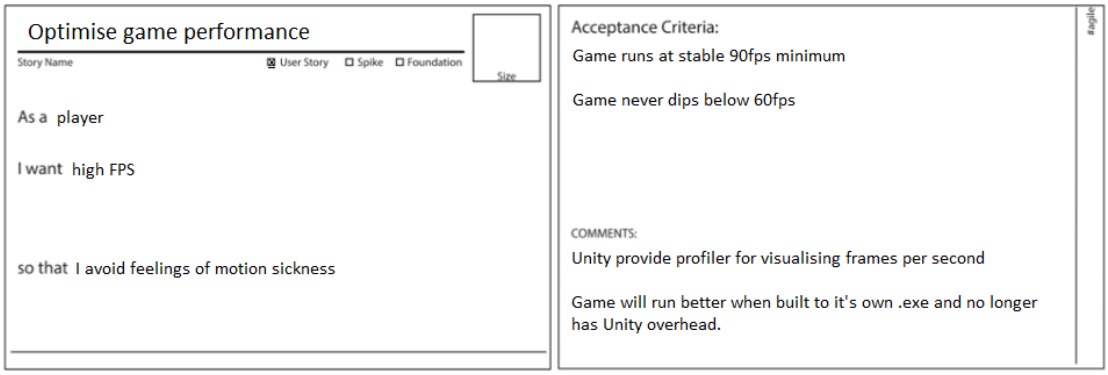
Please include any further relevant information for this section here:

10.3. Scrum User Story Examples

10.3.1. Story 1



10.3.2. Story 2



11. Bibliography

[1] Statistica, "• Virtual reality market size worldwide 2016-2020 | Statistic," 2016. [Online]. Available: https://www.statista.com/statistics/528779/virtual-reality-market-size-worldwide/.

[2] Valve Corporation, "Steam Store," Valve Corporation, 5 04 2016. [Online]. Available: https://store.steampowered.com/app/450390/The_Lab/. [Accessed 24 04 2018].

[3] Owlchemy Labs, "Steam Store," Valve Corporation, 5 04 2016. [Online]. Available: https://store.steampowered.com/app/448280/Job_Simulator/. [Accessed 24 04 2018].

[4] SteamLocomotive.com, "Steam Locomotive Stokers," [Online]. Available: http://www.steamlocomotive.com/appliances/stoker.php.

[5] Valve Corporation, "Valve," [Online]. Available: http://www.valvesoftware.com/.

[6] A. Smith, "Hands On: The Lab, Valve's Portal-Themed VR Games," Rock Paper Shotgun, 2016. [Online]. Available: www.rockpapershotgun.com/2016/03/21/valve-the-lab-vive/. [Accessed 22 04 2018].

[7] Microsoft, "Microsoft HoloLens | The leader in mixed reality technology," [Online]. Available: https://www.microsoft.com/en-gb/hololens.

[8] Unity Technologies, "Unity," [Online]. Available: https://unity3d.com/.

[9] Oculus VR, "Home," Oculus VR, 28 03 2016. [Online]. Available: https://www.oculus.com/. [Accessed 25 04 2018].

[10] Epic Games, "What is unreal engine 4," [Online]. Available: https://www.unrealengine.com/en-US/what-is-unreal-engine-4. [Accessed 25 04 2018].

[11] Alex Andrews, "Scrum Of One: How to Bring Scrum into your One-Person Operation," 2017. [Online]. Available: https://www.raywenderlich.com/162654/scrum-one-bring-scrum-one-person-operation.

[12] HTC Corporation, "Vive Ready Computers," [Online]. Available: https://www.vive.com/uk/ready/. [Accessed 26 04 2018].

[13] PersonGuy. [Online]. Available: https://steamcommunity.com/app/358720/discussions/0/350532536103514259/?ctp=2#c133258092253222557. [Accessed 26 04 2018].

[14] Unity Technlologies, "Rotation and Orientation in Unity," 04 04 2018. [Online]. Available: https://docs.unity3d.com/Manual/QuaternionAndEulerRotationsInUnity.html. [Accessed 29 04 2018].

[15] Unity Technologies, "Home," [Online]. Available: https://unity3d.com/. [Accessed 25 04 2018].

[16] Unity Technologies, "Education," [Online]. Available: https://store.unity.com/education. [Accessed 25 04 2018].

[17] Unity Technologies, "Unity - Unity Pro, Unity Plus and Unity Personal Software Additional Terms," [Online]. Available: unity3d.com/legal/terms-of-service/software. [Accessed 25 04 2018].

[18] TheStoneFox, "VRTK Github," [Online]. Available: https://github.com/thestonefox/VRTK. [Accessed 25 04 2018].

[19] TheStoneFox, "Frequently Asked Questions," [Online]. Available: https://vrtoolkit.readme.io/docs/frequently-asked-questions#section-should-i-be-using-the-unity-asset-store-version-https-www-assetstore-unity3d-com-en-content-64131-or-the-github-master-version-https-github-com-thestonefox-vrtk-of-vrtk-. [Accessed 25 04 2018].

[20] TheStoneFox, “VRTK/LICENSE.md,” 30 08 2017. [Online]. Available: https://github.com/thestonefox/VRTK/blob/master/LICENSE.md. [Accessed 25 04 2018].

[21] ValveSoftware, “steamvr_unity_plugin,” 10 01 2017. [Online]. Available: https://github.com/ValveSoftware/steamvr_unity_plugin. [Accessed 25 04 2018].

[22] ValveSoftware, “LICENSE,” 12 01 2017. [Online]. Available: https://github.com/ValveSoftware/steamvr_unity_plugin/blob/master/LICENSE. [Accessed 25 04 2018].