University of Plymouth

School of Engineering, Computing, and Mathematics

COMP3000
Computing Project
2021/2022

Don’t VReak Out!

Matthew Jonathan Hough

10616194

BSc (Hons) Computing and Game Design

Acknowledgements

James Hayter – Dissertation Supervisor

James Hayter was instrumental in supervising and guiding this dissertation project. Right from the beginning they helped reign back the scope of the project and helped set realistic goals.

Their quality feedback also allowed for the ability to better think through problems and issues encountered throughout the development process.

Unity Technologies – Game Engine

The project was created using the Unity game engine by Unity Technologies. Their excellent game engine allowed this project to go forward. [1]

Valve Corporation – VR Framework

This project would not have been possible without Valve. They have created an amazing development framework for VR projects. Their platform is free and easy to use with quality documentation. [2]

Friends and Family – Project Testers

During development it was essential to test the project and let others play and experiment with the project. Each of these people brought new perspectives, found bugs and gave helpful feedback. Some wished to be anonymous, but those that had no objections to being credited have been listed.

Fiona Lewis, Nathanial Lewis, Freja Clemo, William Butler, Sam Bloxham, Amber Rosewood, Anonymous 1, Anonymous 2.

Abstract

The aim of this project was to create a fun, arcade like VR game in which time management and accuracy mattered. The development of the project was to be done in the Unity Game Engine [1], written in Visual Studio in the C# programming language and built on top of Valve Corporation’s SteamVR framework[2][3].

The target audience is simultaneously broad yet narrow, as of writing ‘VR games’ is a genre in and of itself and is yet to be accurately split into subgenres. Due to this, this project’s target demographic is “VR game enthusiasts”.

The project follows similar games such as Cook-Out, Bobby’s Burgers VR and Job Simulator [4][5][6]. They all consist of an immersive VR environment with plenty of interactable objects all encapsulated by a simple gameplay loop.

The project has met a lot of the initial aims. The player can interact with a wide selection of objects in a variety of different ways. The environment is intuitive to navigate with a consistent visual theme. There is a complete gameplay loop with tasks a player must complete. These tasks did not reach the level of complexity that was planned but are still novel.

Overall, the project has been an excellent learning opportunity. The project did not achieve the level of complexity initially planned but has still been a successful endeavour.

Table of Contents

Word Count

6813 – without appendices, table of contents and word count.

Code Link

Git Hub Link [8.5] - https://github.com/MattHough1999/COMP3000-DontVReakOut

YouTube Walkthrough [11] - https://youtu.be/dxcV5L8T22Y

Introduction

The project; Named “Don’t VReak Out” was designed to be an arcade like Virtual reality game. The name is play on the words “Don’t Freak Out” as a reflection of the originally planned experience. The player would be tested in their accuracy and speed. The gameplay would become more hectic and rushed as players hurried to complete all their assigned tasks.

At time of writing the target audience for VR games is very broad. The genre of “VR Games” accurately describes the demographic but doesn’t help nail a specific audience. In general VR games fall into 3 categories; Rhythm games, Shooter games and “simulator” games. The project was designed to fit into the latter category. Recreating real world tasks in a way that was entertaining to perform in virtual reality. Simulator games typically contain a list of tasks and sub-tasks the player must complete to progress. This was the chosen model for the project.

The project was developed using Unity[1], Visual Studio[2], SteamVR[3] and GitHub[8] with various other tools such as Miro[9] and Google Forms[10] were used during various stages of development.

The project took inspiration from games such as Cook Out[4], Bobby’s Burgers[5] and Job Simulator[6] Games that follow a similar model. Simple, intuitive interactions that can be completed on some level by users at most levels of tech literacy and VR experience.

Background

This project was designed to be a challenge. The author had some experience in Virtual Reality before the project began but had never taken on a proper VR project.

This was the major driving force behind the decision to make a VR project. The author also wanted to try their hand at an exciting new medium of entertainment.

Objectives

An overview of the aims for the project can be found in the abstract and introduction. In this section I have provided a Minimum Viable Product and a Maximally Awesome Product to better bring those points across.

MVP

- Office Environment with 3D models
    - Basic prototype models
- Some form of interaction/interactable objects
    - Some models can be picked up or moved.
- VR functionality
    - Basic Camera and hands
- Basic gameplay loop/features

MAP

- Fully interactable office environment
  - Fully fleshed out office
- Lots of interactable objects and models
  - Every object in the scene interactable
- Fully Functional VR gameplay
  - Solely VR/physics based interactions and gameplay
- Full audio system
  - Audio triggers for interacting with objects

Competition

As mentioned previously there are other games in direct competition with this project. These games were analysed, their gameplay dissected to give an understanding of the features and gameplay that go into such games. Covered here are the noteworthy aspects of the three main competitors to the project, the qualities to hold on to and the ones to avoid.

Cook Out

In Cook out customers approach your sandwich stand and provide you with a sandwich order consisting of sandwich components that you have available to you. To make the sandwich you must chop/slice the components and then assemble the components in the correct order. The accuracy of the placed components also matters and affects the customer satisfaction which contributes to your score. A customer will only wait or a set time before leaving, causing you to fail that level.

Positive Qualities

- Cook Out has a fantastic gameplay loop. It takes ordinarily easy tasks and makes the player perform them in virtual reality at speed.
- The gameplay is varied enough to hold a players attention

Negative Qualities

- The visual theme looks very nice but does not assist the user in ascertaining the function of any given object at a glance.

The project is different to Cook Out because it is set in an office environment rather than a sandwich shop. The major gameplay loop is similarly structured but not designed in the same way.

Bobby’s Burgers

Bobby’s Burgers is similar to Cook Out, The player receives orders to complete and must complete increasingly complex tasks with less and less time to do so.

Positive Qualities

- The visual theme is very consistent. Interactable objects are easy to spot and a player can intuitively know what can and can’t be interacted with.

- Other people can send the player orders adding additional fun to the randomly generated content.
- There are plenty of distractions from the major gameplay loop that add variety and novelty to an otherwise repetitive loop

Negative Qualities

- Some players feel confined within the small play area of the VR scene.

The project differs from Bobby's Burgers for the same reasons it differs from Cook Out. A different environment with a similar but distinct gameplay loop.

Job Simulator

Job simulator has a variety of game modes, most of them involve performing seemingly easy tasks in virtual reality. It lacks the "Order" structure of Cook Out and Bobby's Burgers but contains a much larger variety of objects and scenes that can be interacted with. You can play as a chef, an office worker, a mechanic or a cashier. Each of these modes has more objects and item interactions than the previously mentioned games.

Positive Qualities

- Huge variety of objects and environments to experience and play in
- Mostly consistent visual theme and conventions across game modes

Negative Qualities

- The quantity of objects available can sometimes be overwhelming or intrusive. Slowing or inhibiting gameplay.

Job Simulator is a very different game to this project. As mentioned, Job Simulator has 4 different game modes, all distinct and varied. With so much content it would be impossible to not share some similarities with the project but the two will always be clearly distinct, in visual style as well as functionally.

Jargon Glossary

VR – Short for Virtual Reality

Virtual Reality – An immersive technology where a user is placed inside an environment.

Gameplay Loop – A system within a game that structures how a player will progress

Scene – Another word for a level or environment.

Repo – Short for Repository, A backup of the project in the state it was in at the time.

Push/Pushed – Refers to updating a repository.

Method of Approach

This project’s development can be separated into 5 distinct phases. Each phase was comprised of one or more ‘sprints’ though these were not concrete or written down (as explained in a later section). The Research phase, the planning phase, the initial prototype, the intermediary prototypes and the finalisation phase.

Research and preparation

Since the author had little experience in VR, their approach was initially to research the various approaches to VR development. Before starting the project, the author made the decision to create the project in unity as they were already experienced with the engine. Within Unity there are two primary approaches to VR development. Unity’s In house “XR” development package and Valve Corporation’s “Steam VR” package. The Steam VR package was chosen for compatibility and the quality of its documentation. Steam VR natively handles almost every available Virtual Reality headset whereas unity’s XR plugin would require headset specific code.

Due to the author’s lack of experience in VR development the next step was to familiarise themselves with the differences between VR development and regular development. This was important as much of the development time was expected to be spent learning by doing.

The primary difference is how the user interacts with the scene. In a typical video game, the player is limited in actions they can perform. The use of a keyboard and mouse as the input allows the developer to precisely control where and how a user can interact with any given object in the scene. In a VR game the user expects every object in the scene to look and behave as it does in the real world. As an example, in a traditional video game a cup on a desk has no need to be interactable. The user will think nothing of a static cup as part of the scene. In VR however the user will almost always try to pick the cup up and throw it around, store it inside something else or balance it on themselves/another object. Not being able to do any one of those things will result in dissatisfaction and they will be pulled out of the experience.

A secondary effect of this difference is that every object must have some form of physics attached. This causes its own problems though. Important objects in the scene need to have safety measures and some positional assistance. This is to prevent the user ‘Soft locking’ themselves (Causing a situation/problem that prevents them from progressing despite the game not crashing/or sending them to a game over state). A common method for overcoming these issues is to set a return point for important objects. That way if they travel too far or are somehow destroyed, they spawn back at their return point. This return point can also be used to help a user place an object in the correct place.

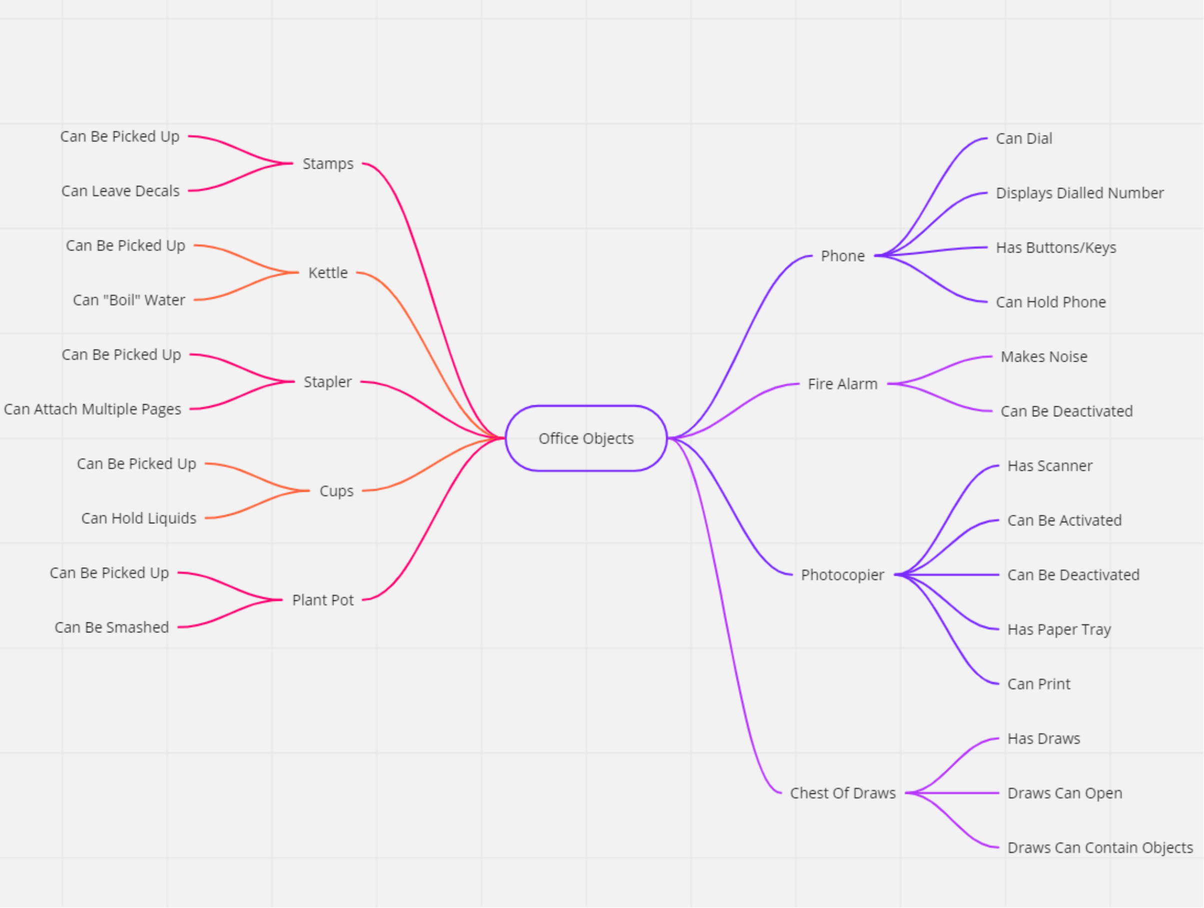
Planning

The planning phase for the project was relatively short and simple. The author looked to VR games that had already been released and specifically investigated their gameplay mechanics and gameplay loops. The most successful games revolved around either A) being some form of rhythm game or B) Recreating a real-life task or series of tasks and extracting novelty from the quirks of VR interaction system.

Rather than attempt to deal with the legal issues surrounding the music and sound effects necessary for a rhythm game the author decided to follow Cook out, Job simulator and similar game's example and recreate typically mundane real-life tasks in VR.

The game was planned to be more similar to Job Simulator, an office environment with plenty of interactable objects and tasks to perform. Adding the complexity of Cook Out's food orders.

The office environment was not formally planned or structured. Instead the author took the idea to friends and family, consulting them regarding objects that they would expect to find in an office. Once a list of objects had been compiled each object was broken down into actions the object could perform and what would be necessary to achieve this behaviour. As an example: A chest of drawers has a number of draws that can open and close. The draw's path can be blocked by other physical objects and the draws can contain objects.



Not planning the office layout allowed for greater experimentation and changes to be made once the models were functional.

Initially the author also wished to make the objects destructible as part of the game. but it was decided that this made the scope too broad.

Initial Prototype

The main purpose of the initial prototype was to gain an understanding of VR development. There were no models in this phase just an empty Unity scene and a character controller.

The author started learning about VR development through experimentation. Adding and investigating with the various scripts and properties that could be given to objects in the scene. The following subheadings will discuss scripts of particular note to the author within the Steam VR package and why they're useful.

Interactable

Interactable is a script that tells the Steam VR package that the game object is to be interacted with. This can take a multitude of forms that will be handled by other scripts (From the package or otherwise) but without this script objects in the scene are not capable of special behaviours involving the VR character controller. The object will still interact with collisions and physics.

Throwable

Throwable is a script that can be assigned to an interactable object that will allow an object to picked up and released. This script should be attached to every loose object in the scene.

Throwable is perfect for a large variety of objects and purposes as it contains many properties and options that control how an object it is attached to behaves. For example: A cup won't feel out of place if you pick it up from any part of it. But picking up tea pot by the spout would look very unnatural.
Similarly, a tennis ball would feel strange if you couldn't throw it around or drop it but a magic sci-fi cube may appear as though it is supposed to remain rigidly in space before/after you touch it. Everything mentioned and more is possible by simply changing parameters on the throwable script.

Linear Drive

Linear Drive is the opposite of the throwable script. It is for objects that you want to be able to move within strict boundaries and limitations. Linear drive has many of the same parameters available to the throwable script regarding attachment points, grab and release triggers.

Linear Drive's most common use is for sliding doors, chests of draws and levers. Objects with a clear starting and end position. The movement does not need to be a straight line, it can be a complex and bendy animation so long as it is a linear motion.

Velocity Estimator

Velocity estimator can be used in combination with the above scripts to make some objects feel more natural/intuitive. It is used to create a more realistic looking arc or trajectory when releasing objects. Without this script objects released by the player will interact with the physics engine as though they had no starting velocity and simply drop to the ground or stop in place. With the script attached and used in the throwable or linear drive's parameters the object will release with the velocity it had when the player released it.

Intermediary Prototype – Office Build

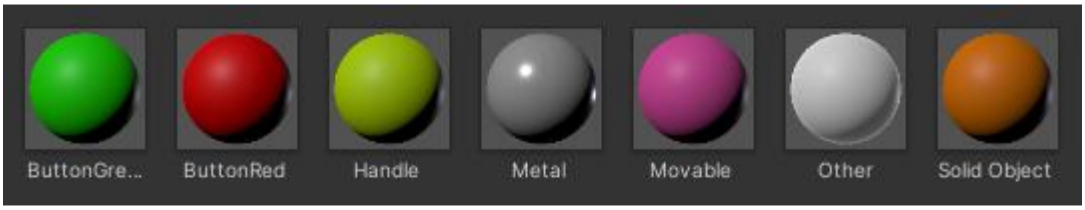
After experimenting with the basics of VR development, the author moved on to creating the office environment for the game. This process began with the necessary models for the objects within the scene.

Visual Theme and Style

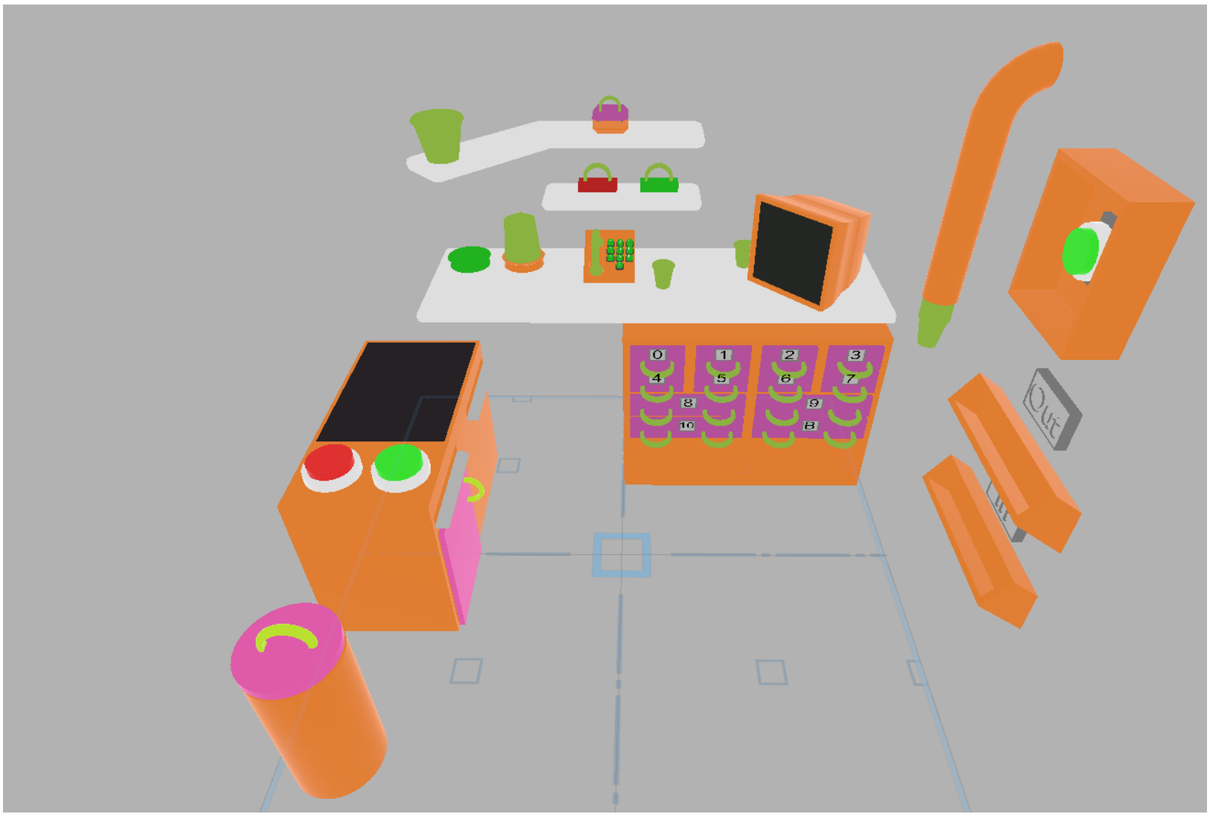
It was in this stage that the office environment started to take shape. Models were created for every planned object. Most of these models didn't change very much throughout the later stages of development as the Author was happy with how they looked and functioned once they were finished. The model that underwent the most change was the model of the desk. It changed many times as the layout of the office changed.

The models were all designed to have a consistent visual theme so that a player could identify the functionality of any object and its components just by looking at it.

There are deliberately no exceptions to this visual theme. This is to maximise its effectiveness. As will become evident in the section regarding testing the visual theme was shown to be very helpful.

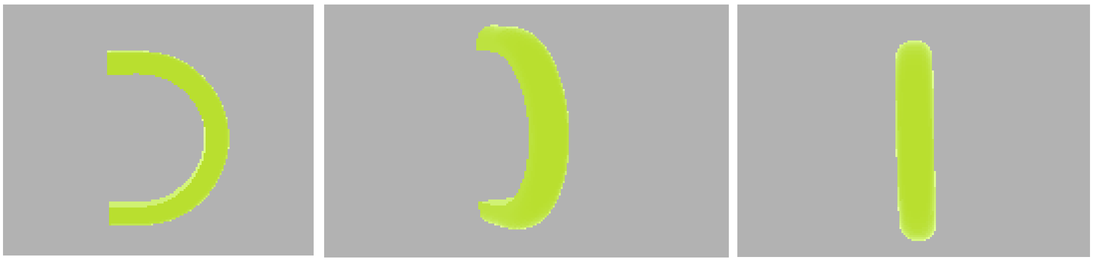


Buttons are either Red or Green with a white base. Anything that can be grabbed or held is given the "Handle" material. This includes Handles, Cups, the Kettle and the Phone. Objects that can move but can't be grabbed are given the "Moveable" Material. Objects such as the Drawers, Bin lid and Photocopier Door. Objects that cannot be moved or grabbed but do have moving components are given the "Solid Object" material and is applied to most other objects in the scene. Objects that cannot be interacted with are marked with "Other". It is applied to the desk, exit door and shelves. Some models have edges or components that do not serve a purpose but add to the aesthetic. These pieces have been given the "Metal" material.



[Figure shows how the materials are used to create the consistent visual theme.]

As well as the colour scheme, another piece of the visual theme was making sure objects that performed the same function appeared physically similar as well as just sharing a material. This is why every handle looks the same, regardless of what it is attached to. The drawers, phone and kettle perform different purposes, but the player understands what parts they can grab intuitively because of the consistent model.



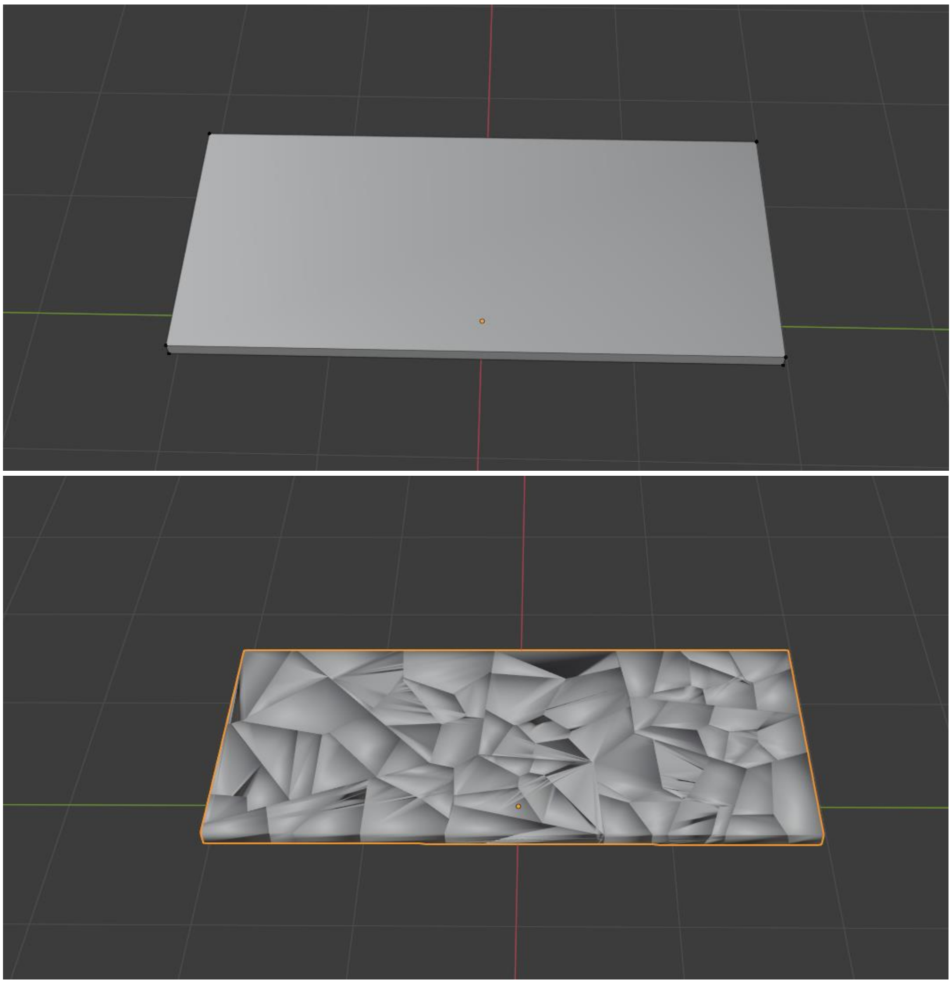
[Figures show the handle prefab]

Movement

Movement in the scene was very simple to implement. The SteamVR package comes with prefab assets for both teleportation and smooth movement. In this project teleportation was the movement system implemented as a user can move around within the environment themselves and teleport to access another area if necessary.

Cut Features

It was during this initial prototype phase that some content was decided to be removed from the scope of the game. The initial plan had included destructible objects. Resources online provided a blender script to break a model up into many randomly shaped pieces. Shown below is a simple cuboid as a demonstration This created a really cool effect in VR and allowed a player to smash objects. However this turned out not to fit in with the rest of the game mechanics and so it was dropped. A clip of how these destriction physics can be found in the appendecies[7]



[Figure Shows how a simple model could be broken into many pieces]

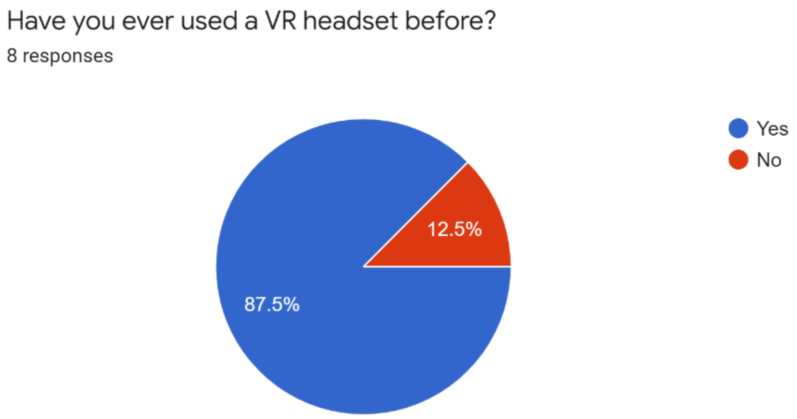
Another feature to be cut was its online functionality. A stretch goal was to be capable of having other people send orders to the player, but this was decided to be far outside scope. This functionality would have been fun, especially during showcase events or players showing the game off to friends. It would help spice up the otherwise simple gameplay loop.

Intermediary Prototype – Initial Testing

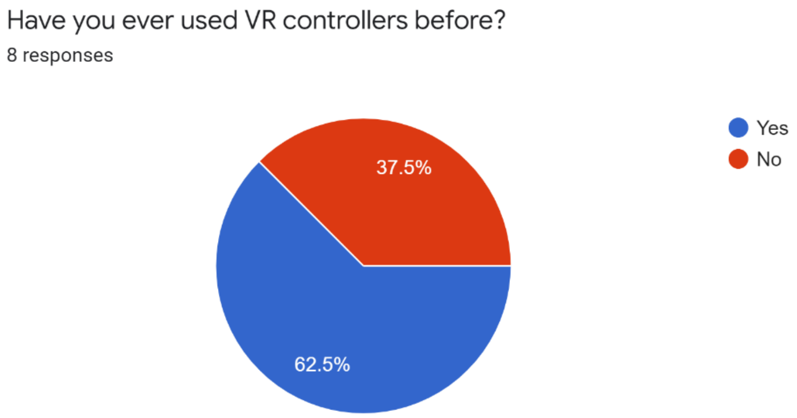
Testing for the intermediary prototype(s) was incredibly informative. The author conducted multiple in person tests of the game throughout development. This initial round of testing was to observe and understand how people interact with a fresh environment.

The testing was done in person and as such the data was collected on paper, it has been collated into graphs for the purpose of discussion.

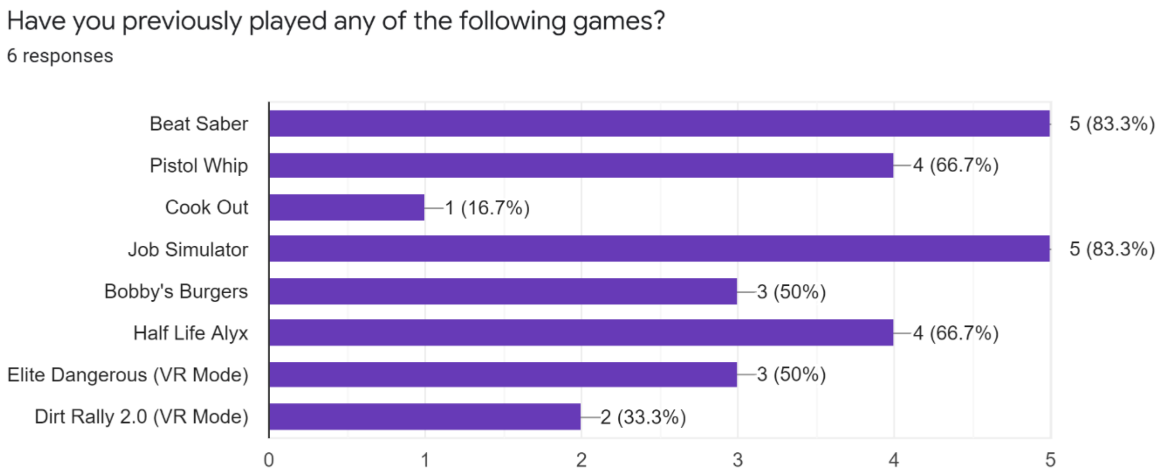
The testers were first asked a series of questions to ascertain their current level of experience with virtual reality and overall tech literacy.



Only one participant had never used a VR headset before. Though the level of experience varied it would have been nice to find some more people who had never experienced virtual reality before to see how complete novices respond to a VR environment.

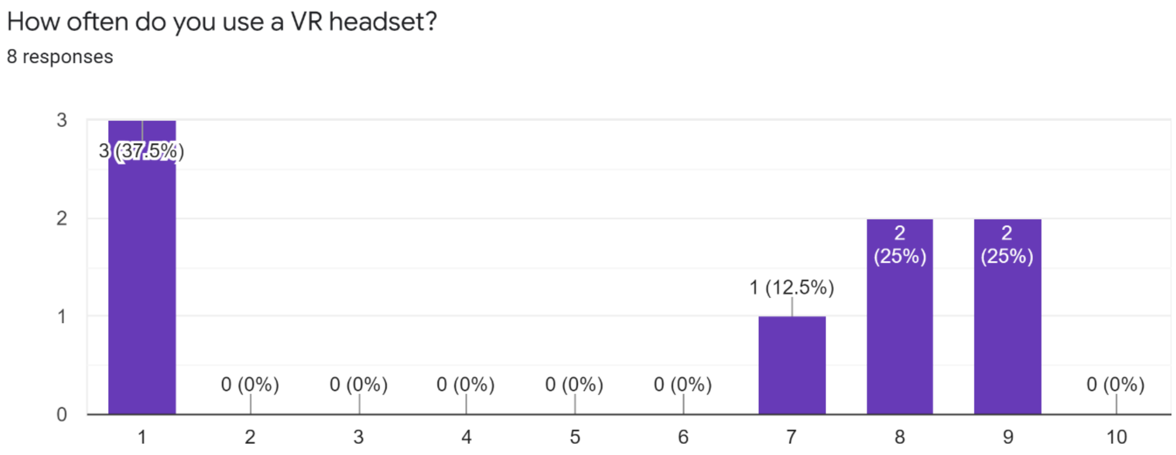


A decent portion of the testers had never used VR controllers before. They had previously experienced devices like the original Oculus Development Kits which were headsets with no controllers. Watching these testers was the most interesting as it took each person a different length of time to adjust to the new controllers/input methods. The objects in the scene and their functions did not seem to affect the length of the adjustment period.

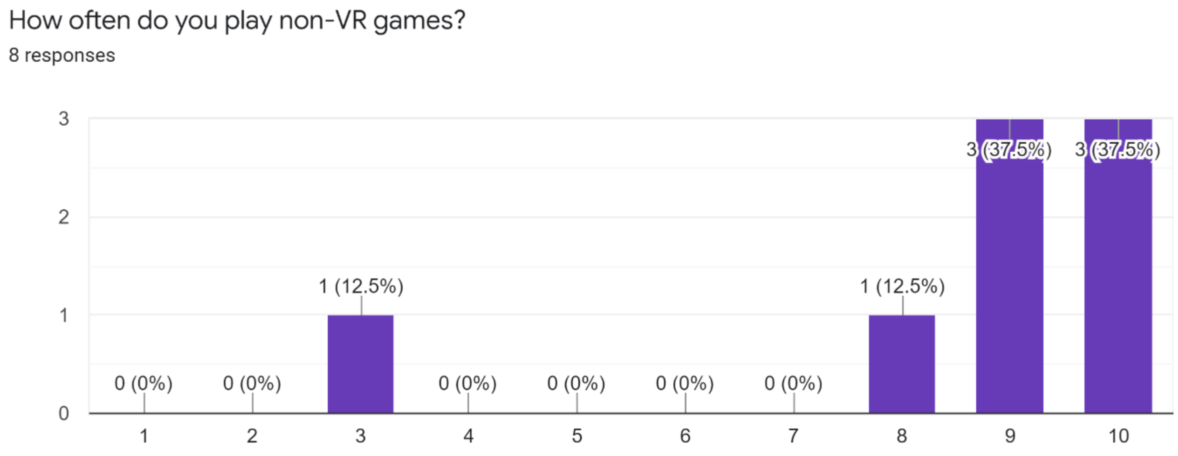

Have you previously played any of the following games?
6 responses

The reason for asking the games people had played before was to understand the type of interactions testers had performed in VR before the test.

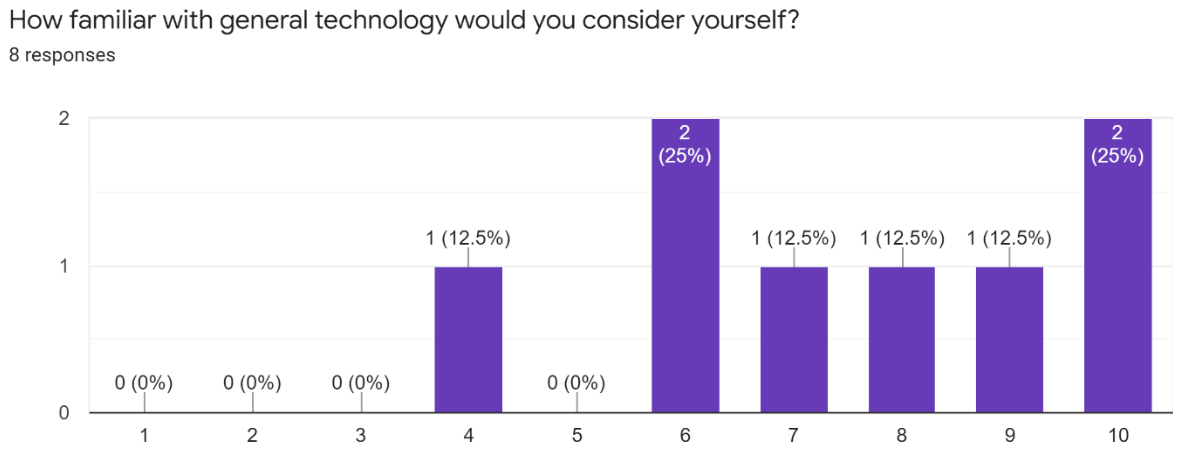
Of most interest were Job simulator and Half Life Alyx. Job Simulator has very similar interactions to the project and experience in that environment definitely appeared to help testers with adapting to the project. Half Life Alyx was also of interest because it has the most complex interaction system known to the author at time of writing. Almost every object can be interacted with in a multitude of ways and behave in much the same way as it would in real life. A tester experienced with HL-Alyx is likely to intuitively understand most forms of VR interactions.


How often do you use a VR headset?
8 responses

This question shows a clear disparity between the level of experience with VR technology. Every tester who owned a VR headset used it very frequently. The question was asked to understand how different experience levels effect how people approach the interactions. The author initially expected those already competent in VR to adjust more quickly. These expectations were fully met with those individuals all almost immediately understanding everything in the scene and how to navigate it while the inexperienced took longer to adapt.

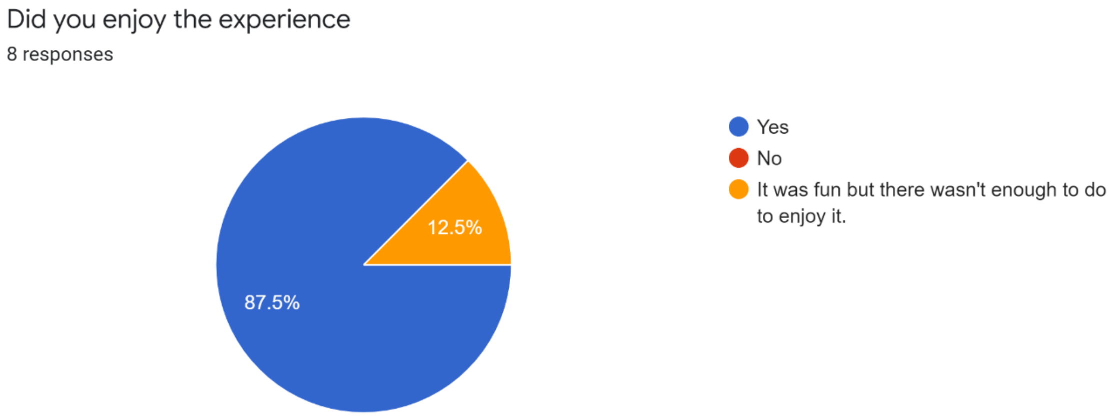


Most participants played games regularly with a single exception. This exception was expected to have the greatest difficulty adjusting to the environment and this was absolutely the case. The individual struggled to understand the input systems at work and had difficulty interacting with the objects in scene.

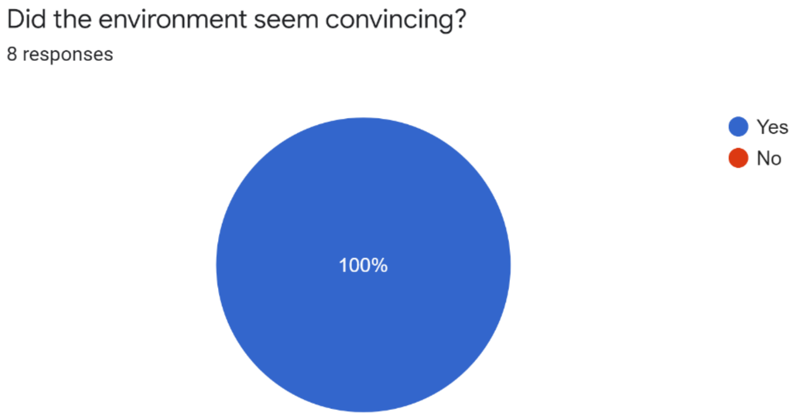


Most of the subjects were or at least considered themselves moderately tech literate. Their relative level of competence with tech did not seem to affect their ability to interact with and adapt to the scenery.

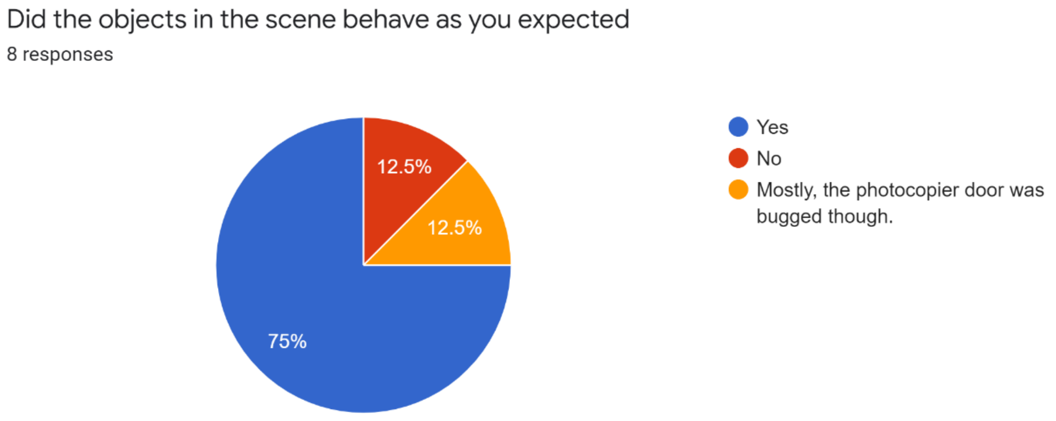
Once these preliminary Questions had been asked each individual then tested project. Experimenting with and getting a feel for the office environment. Participants were assisted into and out of the headset if not confident/comfortable doing it themselves as well as guided through some of the available interactions they could perform in the scene. Most did not require any form of instruction as will be evident in the following data.



Most people said that they enjoyed the experience, though the efficacy of this data might have been affected by the in-person nature of the test. Only one person had a comment other than about their enjoyment which was that there was not enough to do. They clarified to mean there was not enough gameplay however this was not part of the test's purpose.



All of the tested individuals said that they were convinced of the models and their functions. This could be put mostly down to the simple art style and aesthetic helping the testers use their imagination to fill in for the lack of visual detail.

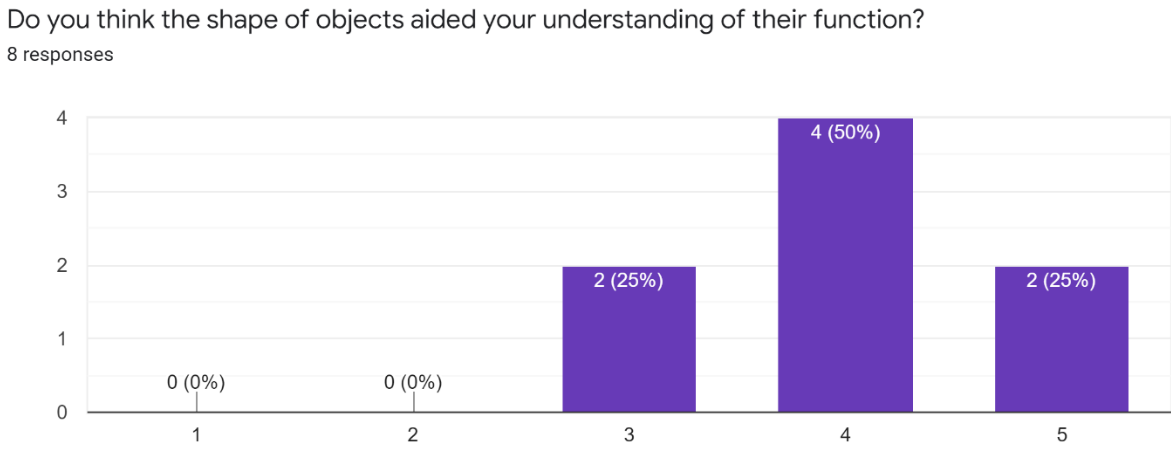


Did the objects in the scene behave as you expected

8 responses

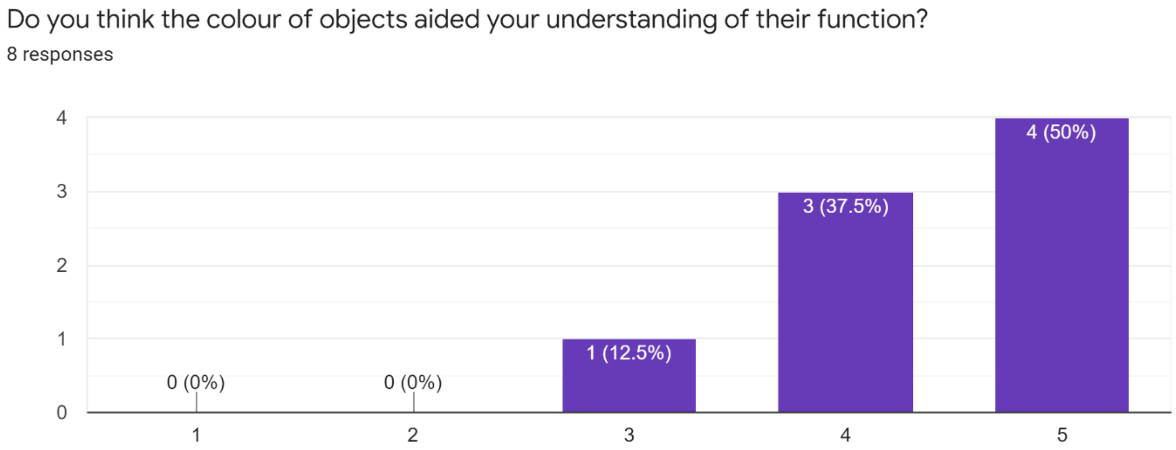
Most people felt that the objects in the scene behaved as they expected them to. Which is positive as this was the purpose of this test. To discover if people understood what to do and how to interact with the environment.

The Comment regarding the bugged photocopier door refers to an unfixed visual bug where the door operates as normal but appears to separate from the door handle.



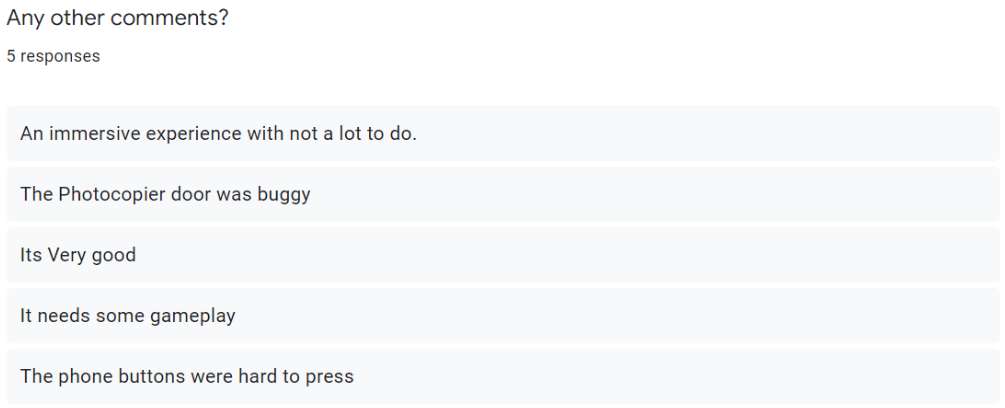
Do you think the shape of objects aided your understanding of their function?

8 responses


Do you think the colour of objects aided your understanding of their function?

8 responses

As mentioned, the author believes firmly that the consistent theme greatly aided the tester's understanding of the environment. This is clearly evident in the data as all individuals responded in a way that suggested it was useful.


Any other comments?

5 responses

An immersive experience with not a lot to do.

The Photocopier door was buggy

Its Very good

It needs some gameplay

The phone buttons were hard to press

Each individual was asked if they had any additional comments to make and had mixed responses. This data isn't particularly useful but did help identify areas people wanted to see being expanded on. Testers mostly seemed to want some form of gameplay loop to become involved.

Intermediary Prototype – Gameplay Mechanics

Satisfied with the office environment and the objects it contained the author moved on to forming the major gameplay loops.

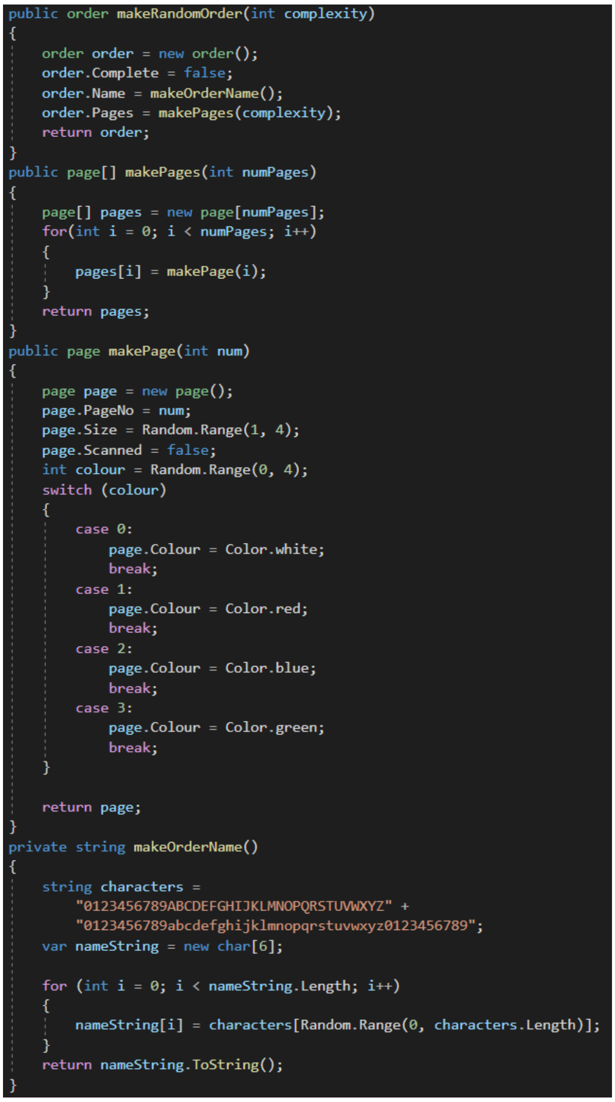
Starting with the initially planned "Order" system which was eventually cut but is worth talking about, not least to understand what went wrong and how it could be implemented in future.

During the planning phase of the project the author identified the gameplay loops of games such as 'Cook Out' consisting of a number of tasks and sub tasks to be

completed within some arbitrary time span.
The author wished to replicate this loop using the components of the office environment.

This was to be accomplished with the use of the photocopier, a player would need to place the correct size and colour of paper on the photocopier, activate the copier to print the appropriate number of copies and then staple them together in the correct order.

Non-Implemented Gameplay



```
public order makeRandomOrder(int complexity)
{
    order order = new order();
    order.Complete = false;
    order.Name = makeOrderName();
    order.Pages = makePages(complexity);
    return order;
}
public page[] makePages(int numPages)
{
    page[] pages = new page[numPages];
    for(int i = 0; i < numPages; i++)
    {
        pages[i] = makePage(i);
    }
    return pages;
}
public page makePage(int num)
{
    page page = new page();
    page.PageNo = num;
    page.Size = Random.Range(1, 4);
    page.Scanned = false;
    int colour = Random.Range(0, 4);
    switch (colour)
    {
        case 0:
            page.Colour = Color.white;
            break;
        case 1:
            page.Colour = Color.red;
            break;
        case 2:
            page.Colour = Color.blue;
            break;
        case 3:
            page.Colour = Color.green;
            break;
    }

    return page;
}
private string makeOrderName()
{
    string characters =
        "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ" +
        "0123456789abcdefghijklmnopqrstuvwxyz0123456789";
    var nameString = new char[6];

    for (int i = 0; i < nameString.Length; i++)
    {
        nameString[i] = characters[Random.Range(0, characters.Length)];
    }
    return nameString.ToString();
}
```

[Figure shows order creation]

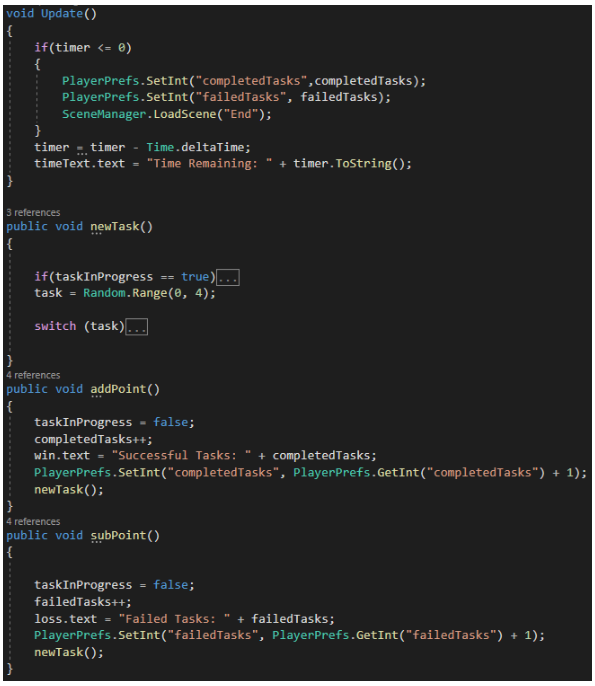
The system for generating these orders was completed and the written code could successfully create a set size of randomly arranged orders. These orders could be accessed and used within code, even across scripts. However, the author was unable to merge this code with gameplay objects. The code for generating the orders can be seen to the left. Page and Order are both structs declared above this point this script. This made them easier to handle and pass to and from scripts. Each order is made of pages, each page has 4 properties: the number of the page in the order; The Page size; Whether the player has scanned the page; and the colour of the page. There were previously additional properties, but these were removed in the debugging process as possible causes of the development issues.

After struggling for a long time with array exceptions with no obvious solutions the feature was dropped and instead the distractions from these work orders became the tasks a player must complete.

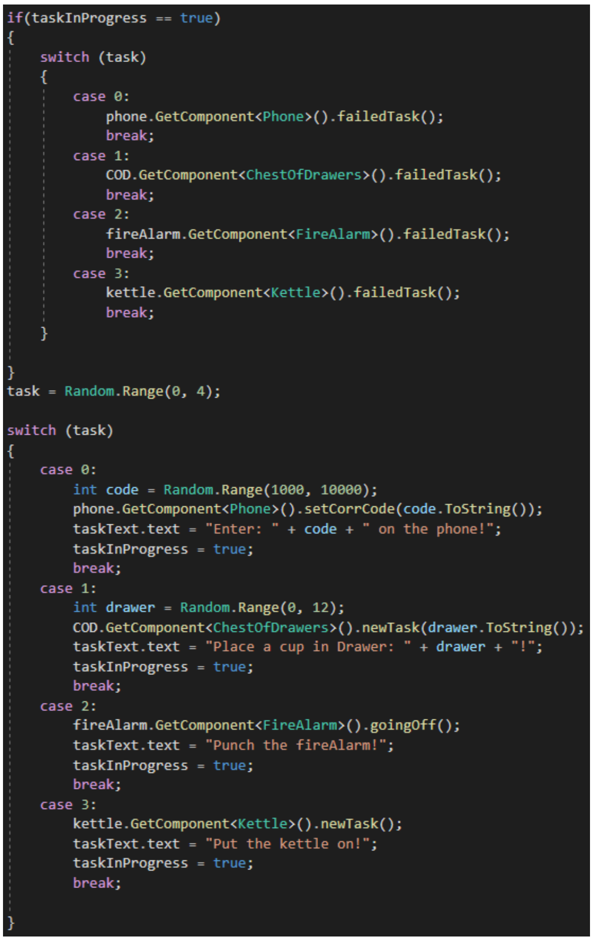
These are less complicated and aren’t linked together but they are functional and somewhat entertaining. In the future the author hopes to find time to solve the issues with the order system and implement them into the main gameplay loop to make the game more fun.

Implemented Gameplay

The following discusses the gameplay successfully implemented into the project. Initially intended as distractions from the primary gameplay loop these tasks are more simple. The script controlling these tasks is just as complex, involving many scripts all communicating with one another.



```
void Update()
{
    if(timer <= 0)
    {
        PlayerPrefs.SetInt("completedTasks",completedTasks);
        PlayerPrefs.SetInt("failedTasks", failedTasks);
        SceneManager.LoadScene("End");
    }
    timer = timer - Time.deltaTime;
    timeText.text = "Time Remaining: " + timer.ToString();
}

3 references
public void newTask()
{

    if(taskInProgress == true)...
    task = Random.Range(0, 4);

    switch (task)...

}
4 references
public void addPoint()
{
    taskInProgress = false;
    completedTasks++;
    win.text = "Successful Tasks: " + completedTasks;
    PlayerPrefs.SetInt("completedTasks", PlayerPrefs.GetInt("completedTasks") + 1);
    newTask();
}
4 references
public void subPoint()
{

    taskInProgress = false;
    failedTasks++;
    loss.text = "Failed Tasks: " + failedTasks;
    PlayerPrefs.SetInt("failedTasks", PlayerPrefs.GetInt("failedTasks") + 1);
    newTask();
}
```



```
if(taskInProgress == true)
{
    switch (task)
    {
        case 0:
            phone.GetComponent<Phone>().failedTask();
            break;
        case 1:
            COD.GetComponent<ChestOfDrawers>().failedTask();
            break;
        case 2:
            fireAlarm.GetComponent<FireAlarm>().failedTask();
            break;
        case 3:
            kettle.GetComponent<Kettle>().failedTask();
            break;
    }

}
task = Random.Range(0, 4);

switch (task)
{
    case 0:
        int code = Random.Range(1000, 10000);
        phone.GetComponent<Phone>().setCorrCode(code.ToString());
        taskText.text = "Enter: " + code + " on the phone!";
        taskInProgress = true;
        break;
    case 1:
        int drawer = Random.Range(0, 12);
        COD.GetComponent<ChestOfDrawers>().newTask(drawer.ToString());
        taskText.text = "Place a cup in Drawer: " + drawer + "!";
        taskInProgress = true;
        break;
    case 2:
        fireAlarm.GetComponent<FireAlarm>().goingOff();
        taskText.text = "Punch the fireAlarm!";
        taskInProgress = true;
        break;
    case 3:
        kettle.GetComponent<Kettle>().newTask();
        taskText.text = "Put the kettle on!";
        taskInProgress = true;
        break;

}
```

A single script decides which tasks to assign the player which then takes those instructions to the scripts attached to the objects involved in those tasks.

This makes the implementation easier to manage but can make debugging harder as tracking down which script is causing the error isn't always as simple as looking at the script where the error occurred. Thankfully this wasn't an issue as the scripts all worked with only logic errors and not fatal ones.

The two figures show the bulk of the script. The script checks that another task is not already active. Fails that task if one is, and then picks a new task.

The first task picks a random code to be entered on the phone. It picks a random number between 1,000 and 10,000 and sends that through to the phone script.
The second task picks a draw in which to place a cup. Sending the selected draw through to the chest of drawers.
The third sets off the fire alarm. Which the player must punch to deactivate.
The final task is to put the kettle on to boil.

At any time the user may press a button in the scene to fail their current task and move on to another one.

Each script attached to a task has a "failedTask" method this is so each task can execute code specific to that task without having to code each of the cases into the overarching script.

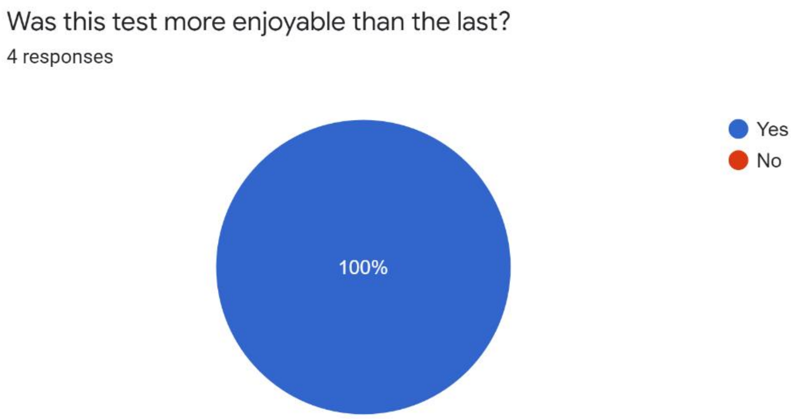
[Figures show the script assigning tasks to the player]

It was during this phase that an issue with the Text assets in use was identified. The Text attached to objects in the scene (such as the letters on the drawers) would show through all other objects. While not a game breaking issue it was noticeable enough that the author decided to fix it later in development. The issue was caused by the text asset not belonging to a canvas object within the scene. As a result, the text asset did not have a sorting layer and therefore could not appear to be behind other objects. To fix this, each Text asset with this problem was simply added a canvas object and put on the correct sorting layer.

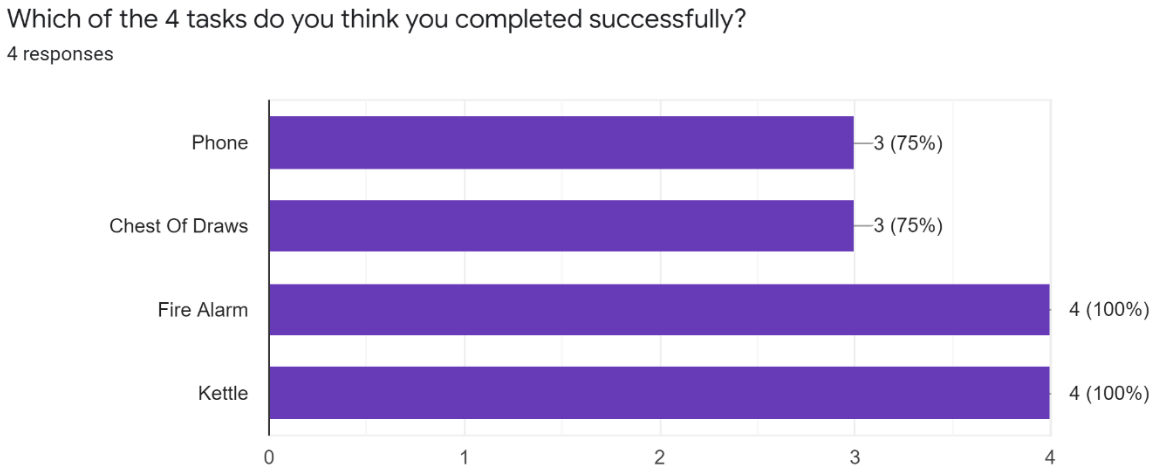
Intermediary Prototype – Gameplay Testing

Once setting up the tasks was completed the project went through a second round of user testing. The purpose of this testing was primarily to search for bugs or other quirks in the project. A secondary purpose was to see what users thought about the tasks themselves. Everyone involved in this set of tests had participated in the previous round so asking about their experience in VR was redundant.

The test was composed of doing the available 4 tasks once, then resetting and doing the 4 tasks again in a different order with different draws and numbers.



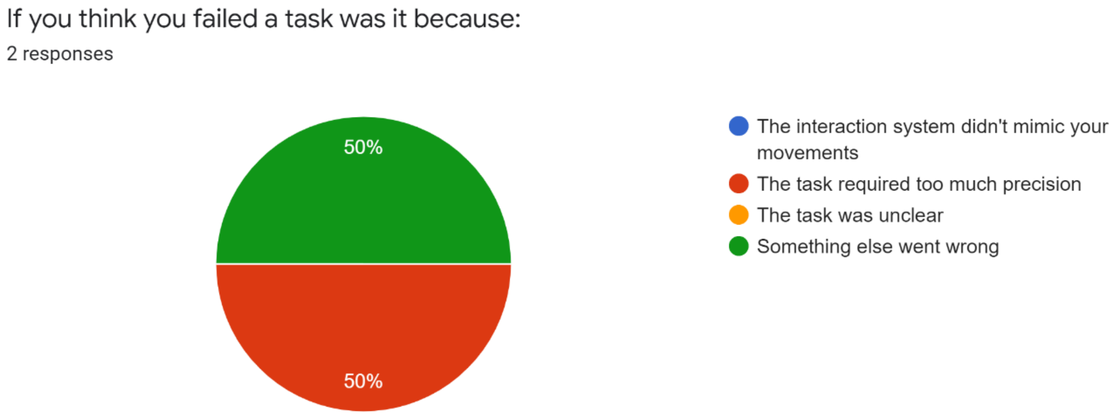
Every participant enjoyed this new version of the project more than the previous. This can mostly be attributed to the presence of gameplay alongside the interactions.



Which of the 4 tasks do you think you completed successfully?

4 responses

Almost everyone completed every task without difficulty. This has both positives and negatives. A lack of challenge means the project should be suitable for anyone to enjoy regardless of technical ability. Though this may mean people with a lot of experience in VR may not enjoy it as much.



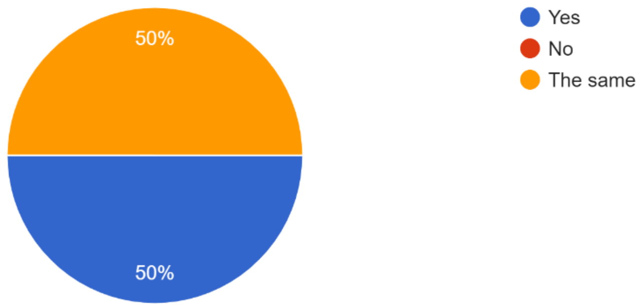
If you think you failed a task was it because:

2 responses

Only two people believed they failed a task on the first time around. In reality three people failed at least one task on their first attempt. Dialling numbers on the phone was failed twice. This may be because of the additional precision required to do so, and this is reflected in the individual who Identified that they had failed to enter the correct number. The “Something else” had to do with the handedness of the person which will be covered in a later question.

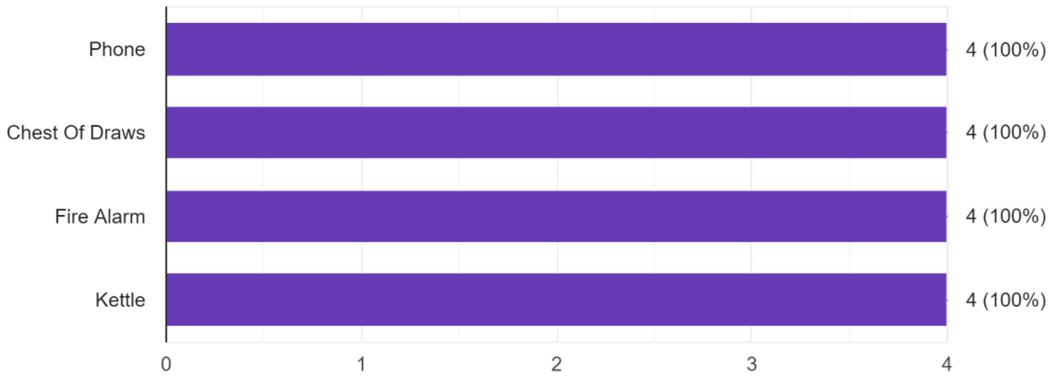
Were the tasks easier to complete the second time around?

4 responses



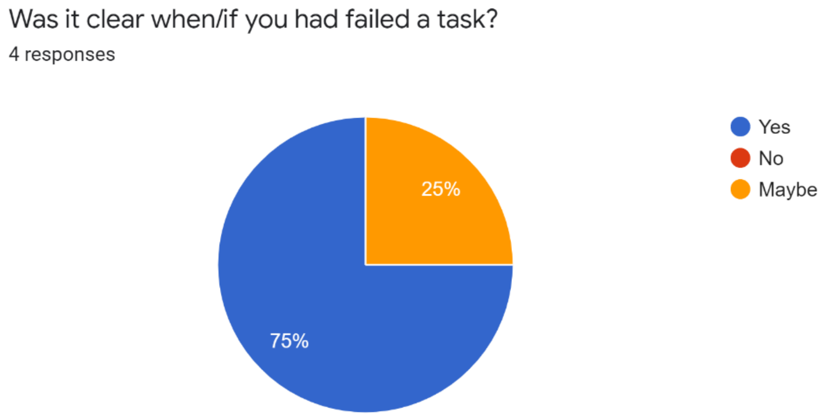
Which of the 4 tasks do you think you completed successfully?

4 responses



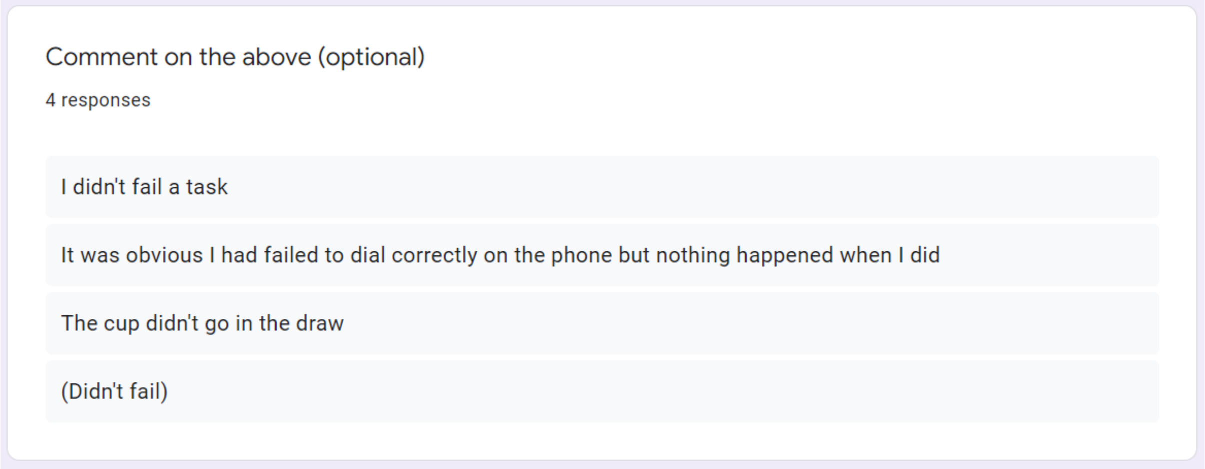
Everyone that participated found the test to be easier or no different the second time they attempted it. This does not necessarily indicate any necessary changes to the project. The second time around they were warmed up and adjusted the headset and environment. To improve the results of this test a third attempt could have been performed but everyone involved successfully completed every task the second time around so the additional data may not have been useful.

If there were more interactions to choose from or the interactions were more complicated a third experiment may have helped find difficult or unrefined areas.



Was it clear when/if you had failed a task?

4 responses



Comment on the above (optional)

4 responses

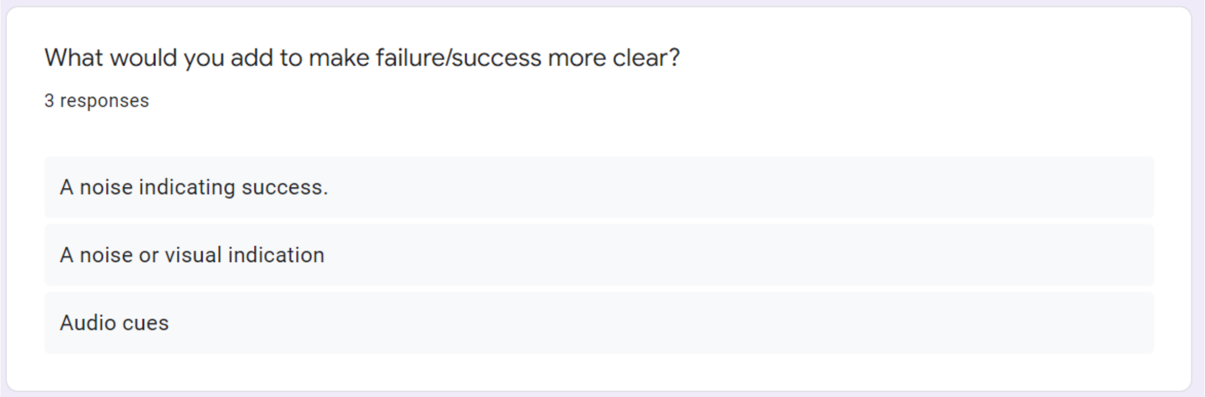
I didn't fail a task

It was obvious I had failed to dial correctly on the phone but nothing happened when I did

The cup didn't go in the draw

(Didn't fail)

This chart is inaccurate, as mentioned one individual was unable to spot that they had failed a task yet decided they could tell when a tasked was failed. In the author’s opinion this is because of the lack of feedback when a task is failed or completed. So other than fully understanding the tasks and being able to see it completed the player had no way of knowing if they had succeeded or failed.



What would you add to make failure/success more clear?

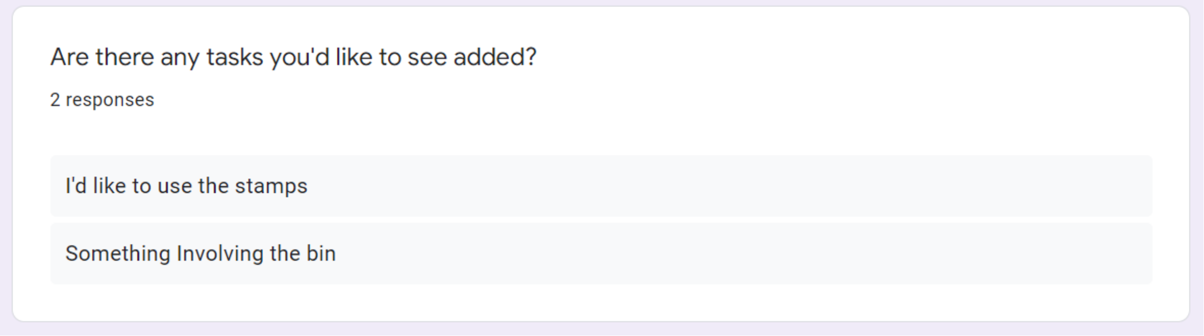
3 responses

A noise indicating success.

A noise or visual indication

Audio cues

In future the project will contain audio triggers to let the player know when they have completed a task successfully or not. This feature was mentioned by all but one of the participants. It is clear that this would improve the experience.



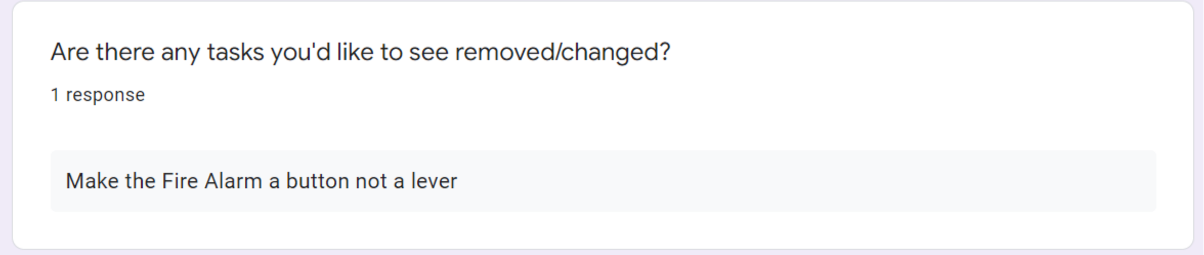
Are there any tasks you'd like to see added?

2 responses

I'd like to use the stamps

Something Involving the bin

The purpose of this question was to determine whether there were any obvious but simple tasks that had been omitted from the project. Stamps and the Bin have been added to the list of features to add but do not appear in the submitted project.

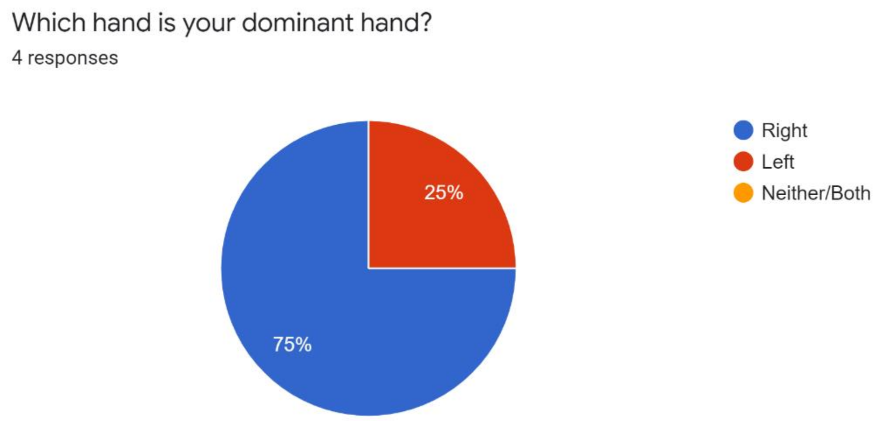


Are there any tasks you'd like to see removed/changed?

1 response

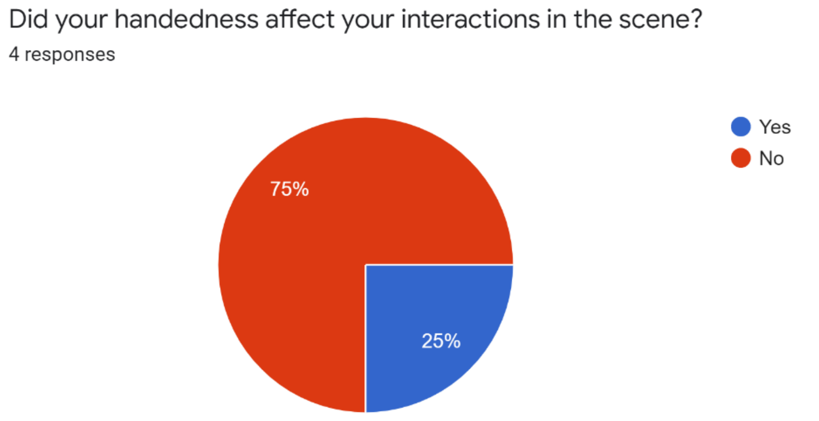
Make the Fire Alarm a button not a lever

The above response was unexpected. The fire alarm was originally a lever with a handle. This was an easy change to make and was made before two of the other participants tested the project. Neither of these testers mentioned the button. It seems as though the button was more satisfying as an interaction than the lever.



Which hand is your dominant hand?

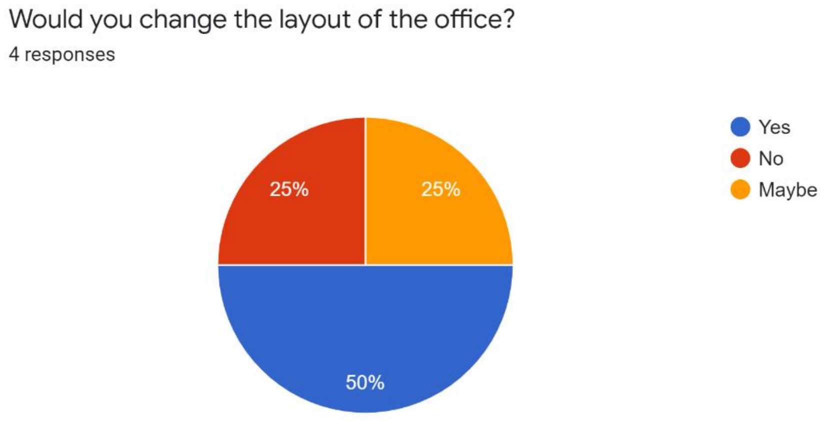
4 responses



Did your handedness affect your interactions in the scene?

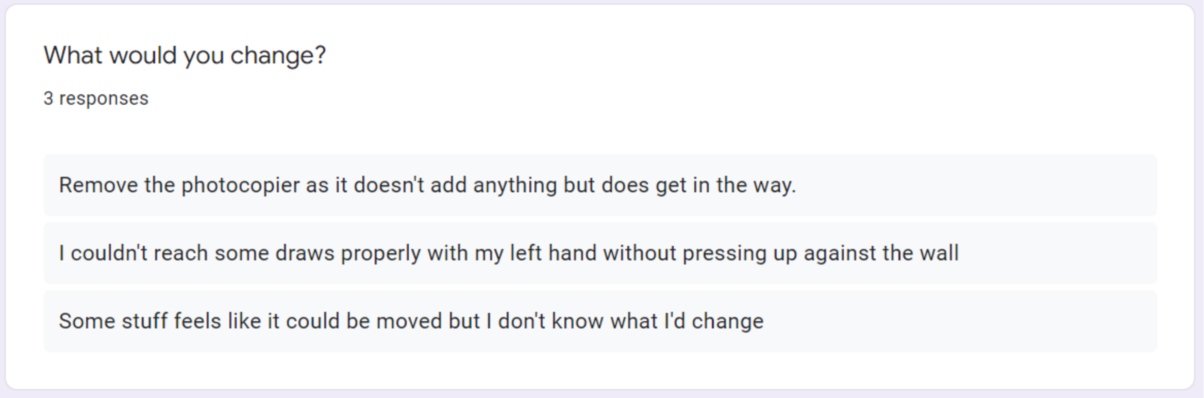
4 responses

The above to questions are important to ask as differently handed people will interact with the environment in different ways. The author is ambidextrous and so the layout and placement of objects in this project does not affect them. It can be a detriment in games that require a specific hand to be used (such as BeatSaber) as the author would naturally use whichever hand was closer to perform the given action. If a specific hand is required then thought is required over pure intuition. For individuals with a dominant hand their handedness can make a difference to their experience. As shown in the test, the left-handed individual struggled with the chest of drawers because there wasn't enough space for them to reach some draws with their left hand. This resulted in them dropping the cup on the floor and not into the drawer. They have of course, just used their non dominant hand but in an already unfamiliar environment this didn't occur to them until they had already failed the task.



Would you change the layout of the office?

4 responses



What would you change?

3 responses

Remove the photocopier as it doesn't add anything but does get in the way.

I couldn't reach some draws properly with my left hand without pressing up against the wall

Some stuff feels like it could be moved but I don't know what I'd change

As you can see in the graph, ¾ of the participants thought that they would change something about the office layout. Moving things around to better suit the handedness of different people has already been covered above. Referring to the photocopier and the TV monitor the individual is correct, the TV screen and photocopier take up a lot of space in the scene but don't currently have any scripting attached to them. They're interactable but the interactions do not serve a purpose.

This second round of testing was very informative. It provided the author with lots to tinker with and to think about. Specifically, the layout of the office. As mentioned, the Fire Alarm was changed to a button. The chest of drawers was also moved to better suit left-handed people, making sure it still suited right-handed individuals. The TV screen and Photocopier remain in the scene and remain unused but were moved to be less intrusive.

Finalisation

Finalising the project primarily consisted of fixing minor quirks in the game. This is the point in the project that the text assets were fixed, and button thresholds were adjusted. There are still some bugs and there is no menu or exit screen but overall, the projects is ready for submission.

Legal and Ethical Issues

The project does not contain any specific legal or ethical issues. Items taken into consideration are covered in the below section. Consent and product availability are the two major factors.

Testing

Each testing participant was asked for explicit written consent to use their names and responses in the data and analysis of said data. Those who did not want their names used still gave permission for data collection and they were appropriately anonymised.

Third Party Software

The use of SteamVR, Unity and Visual Studio are all completely open and freely available to use for a project such as this. Unity and SteamVR have separate rules and licences for enterprises wishing to use their products. These do not apply to the project as it is not for profit and is not being produced by a company making more than $100,000 per year.

Product Availability

The project requires SteamVR to run, as such Steam and SteamVR are both required to be installed on a player's machine. These are both free to download and use. The only requirement is a Steam account.

One item to take into consideration is that SteamVR may not last forever in the state it is in, it may become a paid piece of software or disappear entirely. In this case the project would need to be ported to a different VR framework. This is unlikely to occur but exists as a possibility. Users with SteamVR already installed would not be affected by this issue.

Ethical Considerations

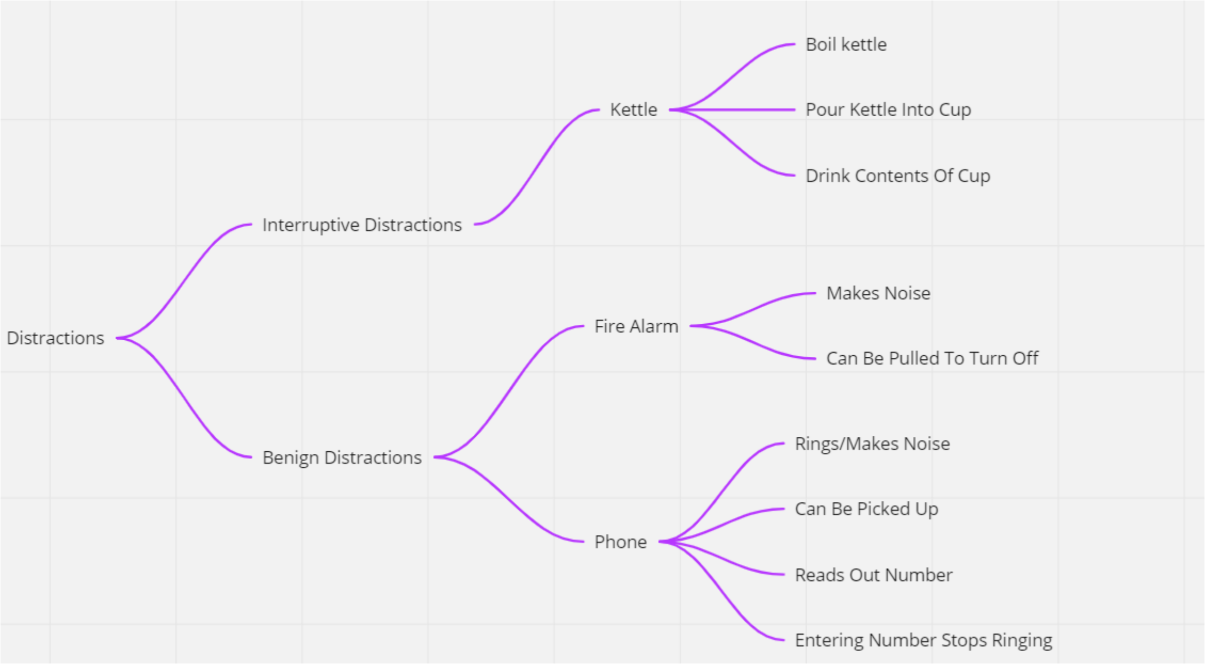
There aren't any important ethical considerations to make regarding the product. Of most note would be ensuring the project does not develop to be too similar to any of the games that inspired it. This is an ethical consideration rather than a legal issue as no copyright would be being infringed however the developers of (or the communities around) the games may react negatively to the similarities.

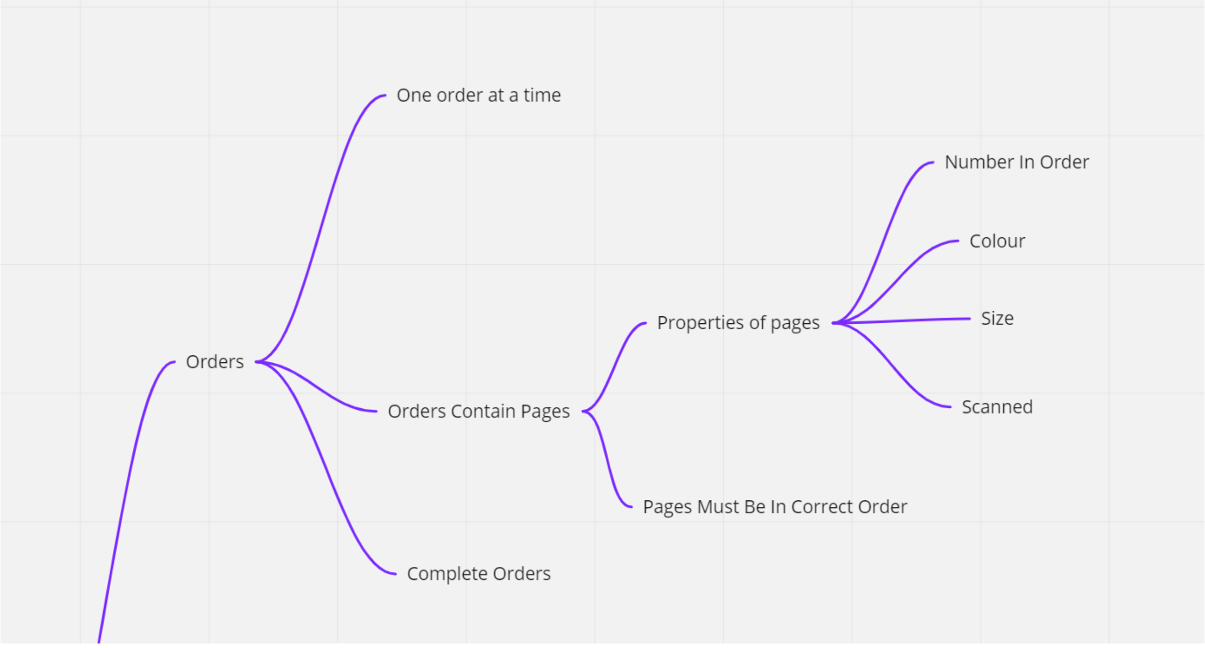
Project Management

There was fairly minimal project management for this project. A GitHub repository was created and used to track the major stages of development with smaller tweaks also pushed to the repo. This could have been pushed to more often to make it a more effective resource for backups and reversing mistakes.

The project did not have an effective or adequate strategy for project management. At the beginning of the project an online software "Jira" was used to create sprints and organise the project. During development access to the associated account was irrevocably lost. Development panning continued with the use of mind maps and word documents. These documents contained all of the features necessary to develop a finished product as well as features that could be added if manageable.

Despite this setback an agile methodology was used throughout development. Testing and updating the project based on the test results but there was no concrete plan or organisation in place after the loss of the initial Jira board.





The above mind maps show the planned features and design for the project. As is evident this is a high-level overview. A more detailed plan along with sprints was on the Jira board.

Project Reflections/Conclusion

The author believes that the project has achieved many of the aims they started out with. The project consists of a complete gameplay loop, in an office environment containing many interactable objects. The MVP is all in place and functional with many of the MAP features implemented, cut or on the ‘To Do’ list

The project does fall short of some of the set goals. Though there is a complete gameplay loop it is not the gameplay loop originally planned. This demonstrates a quality agile approach to development but also demonstrates inadequate planning.

What went well:

- The project has a variety of physics based interactions
- The VR experience is smooth and intuitive
- The Gameplay loop is novel with room for expansion
- The Visual theme is colourful while remaining consistent and intuitive

What Could be Better:

- The Gameplay could be more complex and varied
- The game would be greatly improved with some sounds

What went badly

- The project management was poorly implemented
- Very few pushes to the repo would make reversing mistakes time consuming

Post-Mortem

As discussed above this project was a success. There were flaws in the project management strategy and not all of the desired features were achieved but I learned a lot throughout the entire process.

As previously stated, the initial goal was to have complex orders given to the player similar to the orders given to players in Cook Out. But I am happy with what I have achieved.

In future I would ensure that I have a better system in place for project management before starting development and I would allocate more time to performing proper backups and repository updates

Appendices

[1] – Unity Game Engine, Unity Technologies - https://unity.com/

[2] – Steam VR, Valve Corporation - https://store.steampowered.com/app/250820/SteamVR/

[3] – Visual Studio - https://visualstudio.microsoft.com/

[4] – Cook Out VR – A Sandwich Tale - https://store.steampowered.com/app/1523720/CookOut/

[5] – Bobby's Burgers - https://www.viveport.com/dc071866-2ad3-48ff-aa2e-532de7a16b1b

[6] – Job Simulator - https://store.steampowered.com/app/448280/Job_Simulator/

[7] – Demonstration of the cut destruction physics - https://youtu.be/vXqG5Q-XKks

[8] – GitHub - https://github.com/

[9] – Miro - https://miro.com/

[10] - Google Forms - https://www.google.co.uk/forms/about/

[11] - YouTube Walkthrough - https://youtu.be/dxcV5L8T22Y

[8.5] - Code Repo - https://github.com/MattHough1999/COMP3000-DontVReakOut