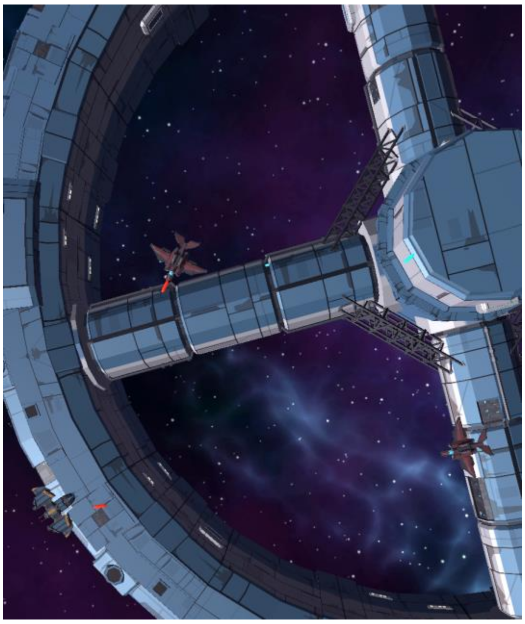


LIBRA IMPERIUS

A Final Year Project

Josh W
Computer Science with Artificial Intelligence

Statement of Originality

This report is submitted as part requirement for the degree of Computer Science and Artificial Intelligence at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Summary

The project showcases my ability and to learn additional skills that I can take into employment after finishing my degree. I chose to make a game as games are something that I spend a lot of my free time playing. I have a lot of experience in playing games and so it felt natural to want to blend my university knowledge with my game understanding. I have limited experience in video game design principles and so this will hopefully expand my skillset in something I have not had the opportunity to do so during the duration of the degree. I have gone through the whole development process from planning, to developing, to implementing, to user testing and finally a critical evaluation of my game. The game itself is a demo to showcase different components of video game programming, such as pathfinding, AI state behaviour, scene transitions, combat, movement, asset management, sound effects, the user interface, and user experience.

Table of Contents

Table of Figures

Introduction

Below I introduce the project, its initial objectives, motivations, and problems that I faced during its development, as well how I solved those problems.

Objectives

This project's objective was to create and evaluate a 2.5D Computer Game. This is achieved by using Unity's 2D game engine (Unity, 2022) and running on Windows 10.

The focus of the game is for players to travel through a group of occupied space systems to remove pirates who are operating illegally, commandeering system stations. Inside each system there are a collection of non-player ships that are both pirate and civilian. These have AI behaviours to drive them and will respond and interact with one another and with the player.

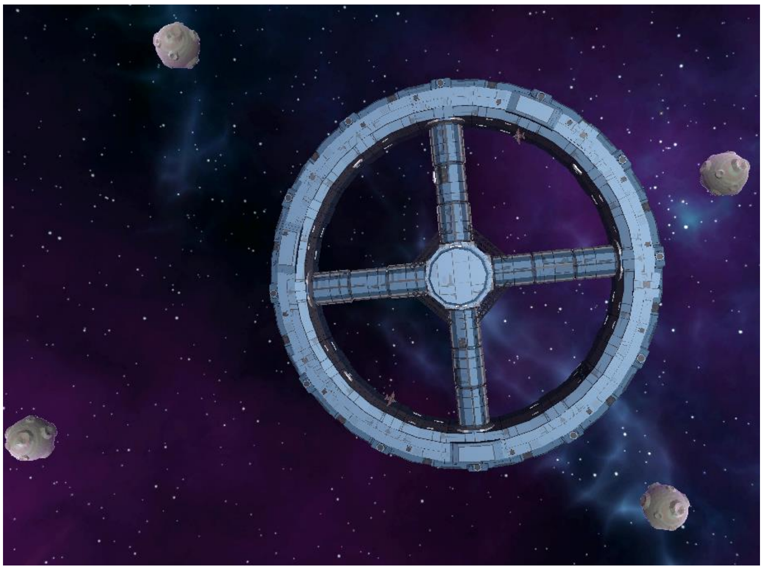


Figure 1 Demo level

There are also asteroids inside of the systems which contain resources that the player can sell off to stations. These funds can be used to repair the ship should there be damage or to upgrade to bigger and better ships. Players will be able to save their progress at friendly stations so that it can be quit and resumed at a later point.

Motivations

I have chosen to create this project to both expand on current knowledge and to demonstrate what has already been learnt throughout the time spent on my course. I feel that my skills such as project management, object-oriented programming, and user interaction, have all been further improved from this project. Completing this project has allowed me to pick up new knowledge in the gaming industry, an industry I wish to eventually be a part of and as such will use projects such as this to showcase the skills I have acquired.

Development Problems

During development, there were parts of the project that I found much harder than others, areas where my skills were not as strong and so I looked for a better way to complete them such as getting an asset pack or finding software to create sound effects for me.

The Engagement Problem

How fun a game is to play determines whether a player will continue to play a game or will look for another game to play. While there is no definitive definition of what makes a game fun, there are ways to make it so that the player can customise the game in a way that makes their fun factor much higher. This can include changes in difficulty and player choice. As described in Star Wars Galaxies' Raph Koster's book "a Theory of Fun for Game Design", Raph talks about how players are drawn to play with their strengths (Koster, 2013). Where some players enjoy the creativity of an open world and limitless opportunity, this is a negative factor for people who prefer a solid structure where they are given direction. Minecraft (Mojang, 2022) is a great example of a game where player choice drives the story. There are no objectives except those that the player defines. The introduction of additional content from third-party content creators, such as the Create mod for Minecraft adds constant new features to the game and allows the game to continue offering new experiences to satisfy the biological desire to learn. Players often look for the most efficient route to complete an objective which may not always be the most enjoyable to everyone, so it is important to make sure there are lots of things to do so that there is a variety of actions. A great example of this is speed running. Speed running is a competitive form of playing a game where the goal is to complete an objective in the shortest possible time, often by any means necessary. Once a player speed runs a game, they record their run and their time before publishing it to a speed running website (Elo Entertainment Inc., 2022). It is a worthwhile endeavour to survey a group of users on how fun they found the game once they have played it for some time and what they find fun in games.

Skill Requirements

This project required a variety of skills. Completing a single-person project often takes many months or years of development to make a fully marketable game. These games often come from individuals with experience in the field with prior knowledge under their belt. As this was my first major work, I was not as versed in these skills and as such broke down tasks to their simpler terms and found ways to create them. Most games use simplistic assets as either their main theme, or to prototype the game so that they can create a demo that is playable to get some feedback from testers. There are 4 main roles in a game:

- Writing Skills – storytelling, dialogue, lore
- Sound Design – music, ambient, effects
- Visual Design – assets, animations, textures
- Programming – code, environment

As I was only really experienced in writing skills and programming I was keen to delve into the sound and visual aspects of creating a game. Creating a compelling storyline and meaningful dialogue would form a solid foundation for the narrative and form a purpose for the player. A demo built however will not need to go heavily into the storyline as it is meant more for creating the grounds of the game, instead of a full build of it. The same can be said for the sound design. A use of an asset pack will massively help in reducing the development time as I have no experience in it and see it as something that could change later in development so should avoid spending too long on it for the demo build.

Writing skills

Writing dialogue for a game takes place toward the end of the project as the game world needs to be constructed for that dialogue and lore to be put in place. While the whole story could be storyboarded, for a demo build, this won't really be necessary. There is also the worry that if too much story is added for a demo, a lot of time would be spent checking consistency and player connection to the story so that the player doesn't burn out.

Sound Design

Sound Design is made up of lots of small components from the musical score to the sound effects. This can take a long time and requires the use of software such as Audacity, Ocenaudio (Audacity, 2022) or BFXR (Lavelle, 2023) to record sounds via pitch changes. I had no previous experience in sound design, so I was excited to create sound effects and decided to create minimalistic sounds that are simplistic, and recorded sound effects where needed, but still suit the needs of the game.

Visual Design

Visual Design is important as the aesthetics of a game will very quickly turn people toward or away from it. For example, an 8-bit design could easily dissuade some players who prefer a more natural/realistic look in a game such as Ghost of Tsushima (shown in the figure below) (Sony Interactive Entertainment, 2023). This is the reason I decided to go for a low poly style, which is soft on the eyes and from a distance (where the camera will be) still looks visually pleasing.

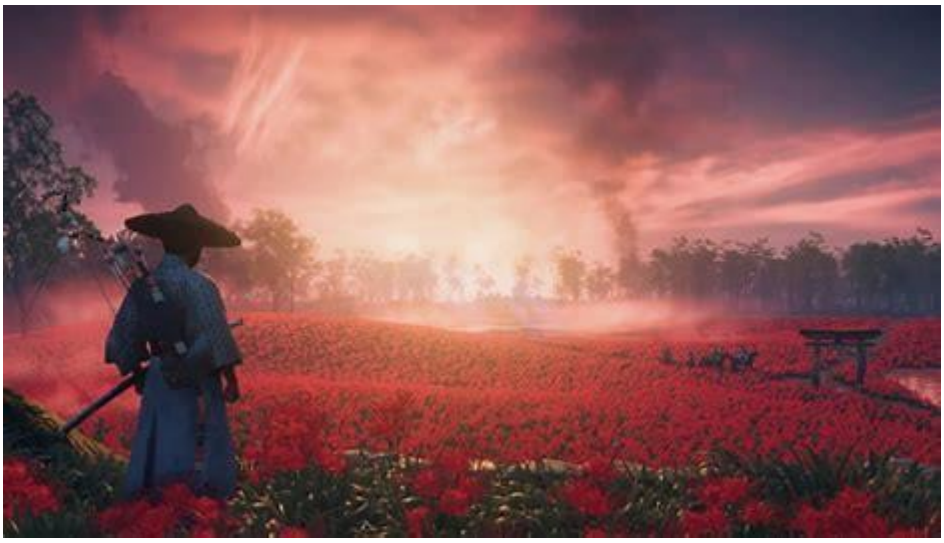


Figure 2 Ghost of Tsushima

Programming

Programming is the area that I felt most comfortable with. It forms the crux of the game and allows systems to interact with one another. One difficulty I faced with programming is compatibility. It was imperative that as the code base was built, different scripts could interact with one another easily and to avoid hard coding in solutions, such as defining an input to be only a single key, rather than allowing the key to be remapped by the user to their preferred controls. This can be applied to almost anything and as such, it is necessary to ensure that changes can be made later during production which means old code does not have to be completely rewritten to be compatible with new code.

Programming also includes other aspects such as what engine the game runs on, or how progression is saved. For the engine, I use a pre-built engine, as custom engines often take large software teams and large amounts of time to handcraft, as such I use the Unity 3D engine. I intended to save the player’s progress using a binary formatter using Unity’s built-in input and output functions.

The user interface is a concept which contains parts of both visual design and programming. Programming allows smooth transitions and the backbone while the visual design ensures the menu icons, and the interfaces are fitting with the theme and are minimalistic enough that the player does not need to try to see past them to play the game.

Report Structure

The report is broken down into several chapters.

The professional considerations will go through the BCS Code of Conduct and how the code of conducts applies to my project. I will then go into detail about the objectives that I considered mandatory for a build of the game to be playable, and the extension objectives that I intended to complete if I had time to implement them. After this, I discuss works that are like my own and are inspirations for my work. I then go on to discuss the designing and implementation of the various features that made up my game, also discussing the development cycles. After this, I talk about my introduction of the demo to play testers and gather feedback on what they thought about the game, features they would like to see added, thoughts on proposed features, bugs found, and general opinions about the game. Finally, I perform a critical evaluation of my progress on the project.

Professional Considerations

Below is each section of the BCS Code of Conduct and the response.

1.1 have due regard for public health, privacy, security and well-being of others and the environment.

The game will be suitable for all players and the only data that will be collected is not personal to the user and simply acts as the storage of the player's location in the game world and the information about the progress they have made.

1.2 have due regard for the legitimate rights of third parties.

Where third-party content is used inside the project it will be free to use and the creator credited for their work.

1.3 conduct your professional activities without discrimination on the grounds of sex, sexual orientation, marital status, nationality, colour, race, ethnic origin, religion, age or disability, or of any other condition or requirement.

The game will have some dialogue between characters but there will be no promotion or exercise of discrimination of any kind. The player character will be featureless to not force features on the player.

1.4 promote equal access to the benefits of IT and seek to promote the inclusion of all sectors in society wherever opportunities arise.

Attempts will be made to accommodate all users.

2.1 Only undertake to do work or provide a service that is within your professional competence.

My supervisor and I agree that this work is within my professional competence.

2.2 NOT claim any level of competence that you do not possess.

My supervisor and I agree that this work is within my professional competence.

2.3 develop your professional knowledge, skills, and competence on a continuing basis, maintaining awareness of technological developments, procedures, and standards that are relevant to your field.

Throughout this project I will learn new skills to improve ability and competence.

2.4 ensure that you have the knowledge and understanding of Legislation* and that you comply with such Legislation, in carrying out your professional responsibilities.

2.5 respect and value alternative viewpoints and, seek, accept, and offer honest criticisms of work.

User surveys towards the end of the project will give the chance to view alternate viewpoints and give honest anonymous feedback.

2.7 reject and will not make any offer of bribery or unethical inducement.

There will be no bribery or unethical inducement at any point, whether that is during development, during user testing/feedback, or after.

3.4 NOT disclose or authorise to be disclosed, or use for personal gain, or to benefit a third party, confidential information except with the permission of your Relevant Authority, or as required by Legislation.

There is no confidential information so it will be impossible to disclose such information.

Requirements Analysis

The requirements of the project are categorized into two separate sections, mandatory objectives, and extension objectives. Mandatory objectives are goals that were considered necessary for the game to be considered playable (a demo build). Extension objectives are goals that add to the user experience and create a more polished experience. Extension objectives are not required for the demo build (the mandatory objectives) but will later become mandatory objectives for a full release of the game.

Mandatory Objectives

- The game runs on Windows 10.
- The player will be able to move in any lateral direction.
- The game is viewed from a top-down camera, centred on the player.
- The player will be able to fire weapons.
- The game will have hostile ships (Pirates).
- The game will have stations.
- The game will feature a hit-points system.
- The game will feature environmental assets such as asteroids.
- The game has a winnable state.
- The game has a narrative storyline.
- There will be sound effects for player ships.
- There will be sound effects for hostile ships.

Extension Objectives

- There will be an intro sequence to familiarise the player.
- The game should feature save points.
- There should be at least 5 station types.
- There should be at least 5 ship types.
- There should be a difficulty setting.
- There should be achievements for exceptional feats.
- The game will try to accommodate people who have hearing disabilities (playable without audio).
- The game will try to accommodate people who have visual disabilities (easily readable text, colour-blind correctness).
- The game will try to accommodate people who have motor disabilities (Simplistic/few controls, remappable keys).
- The game will feature an honour system.
- The game will feature a faction system.
- The game will feature at least 3 systems.
- The game will get more difficult as they travel between systems.
- The player can restore their hit points.

Similar Works

While there are countless space themed games, there are some that are similar to my project.

Cosmoteer: Starship Architect and Commander

Cosmoteer (Walternate Realities, 2022) is a game where ships are made up of components, such as armour pieces, command modules, weapons and more. The player constructs ships by assembling these modules. The modules themselves are made up from a variety of components such as steel, electronics and more. This constant need to expand the ship encourages the player to attack hostile ships and to trade with the many stations in each system to increase the size of their ship and their power. There are multiple factions in the game, each with their own style of ships.

This game is like my own project in that the camera features a similar top-down camera, the ability to fight hostile spaceships, firing weapons, stations, and multiple factions.

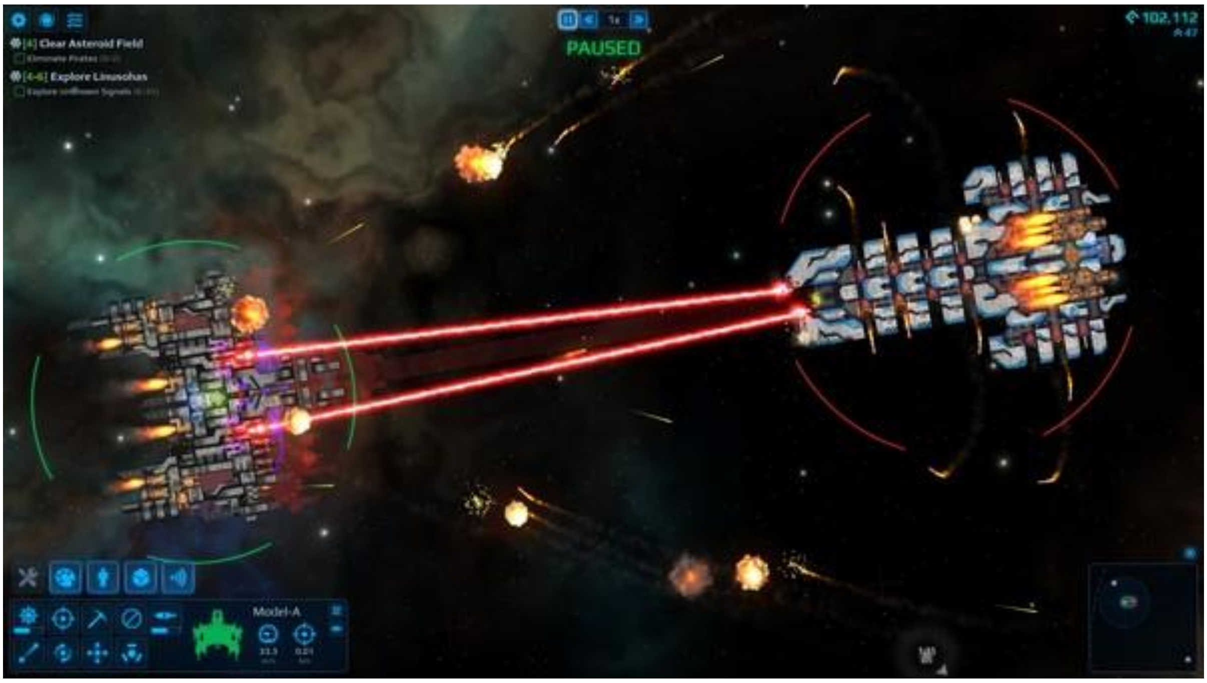


Figure 3 Cosmoteer

Everspace

Everspace (ROCKFISH Games, 2022) is a rogue-like space game where the player enters the world with no knowledge of it. The player picks a spaceship to take into the next run which are all varieties of fighters. Each stage consists of several smaller levels where the world is filled with asteroids, rogue ships, a spaceship company known as G&B which the player can repair their ship at or trade resources. The player has the option at any point to engage this neutral faction to steal resources. This will however cause them to turn hostile for some time.

While my game is not a rogue-like, it does feature ship to ship combat and a storyline to work through. The story in Everspace becomes more apparent while the player explores new sectors or dies and once again tries to reach the end objective.

Figure 4

Drone Swarm

Drone Swarm (Still Alive Studios, 2022) is a game where humanity has lost their home and the player must control a swarm of drones around a ship containing the rest of humanity through many sectors, learning new skills such as a defensive stance or different attacks to use against hostile ships. As the systems progress and the player unlocks new skills, the hostile ships get harder, with unique mechanics to increase the difficulty, such as not being able to be hit from the front due to an energy shield and much more.

This game is alike to my own project as the player will learn more about the story and gaining new skills as they clear out each sector, as increasingly difficulty enemies will try to take out the player.

Figure 5 Drone Swarm

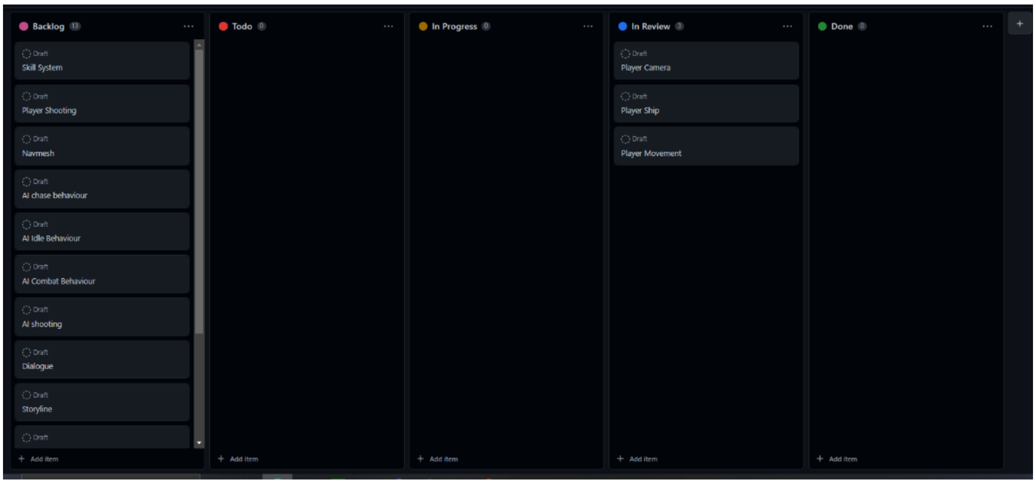
Design And Implementation

This chapter will break down the design process and how objectives were met and implemented.

The design and implementation of the project was split into two-week sprints where the progress of the project was discussed with my supervisor, objectives were reviewed, and the next sprints objectives were identified. This use of agile development allowed me and my supervisor to ensure that the project was making steady progress each sprint, setting objectives that were achievable, and were not overly ambitious for that sprint. Overall, the use of agile development encouraged and rewarded forward planning.

Sprint One

This first sprint was used to set up workspace and get a simple player controller working with a camera system.



Backlog 13

Draft
Skill System

Draft
Player Shooting

Draft
Navmesh

Draft
AI chase behaviour

Draft
AI Idle Behaviour

Draft
AI Combat Behaviour

Draft
AI shooting

Draft
Dialogue

Draft
Storyline

Draft

Todo 0

In Progress 0

In Review 3

Draft
Player Camera

Draft
Player Ship

Draft
Player Movement

Done 0

Figure 6 Sprint one project board

The Player

The player is where the experience is cantered for the user and so it was important to start here and move outward from this point. The first step in designing the player was to first have a model to work with. For this purpose, I retrieved some assets from an asset pack on the unity store.

Camera Controls

For the player camera, I wanted to have the player in the centre of the screen but as the players mouse moved towards the edge of the screen the screen should shift toward it, leaving the player in view but at the same time on the edge of the screen. This was first attempted by getting the position of the mouse on screen and slowly moving the camera between the ship and the mouse, such that the player could see more in the direction of the mouse. However, this had an undesirable effect on the controls and as such I decided to take a different approach. I instead opted for an empty game object that was placed a small distance in front of the ship but as a child of the player ship so that the camera could instead focus on that object, creating the illusion of looking toward the curser.

I decided that the player should control the direction of the ship by their curser such that if their curser was on the right-hand side of the screen, the ship (originally started facing the top of the screen) would rotate 90 degrees to the right and face the curser. This movement felt natural to use and meant that the curser is the focal point of the player's perspective.

Player Controls

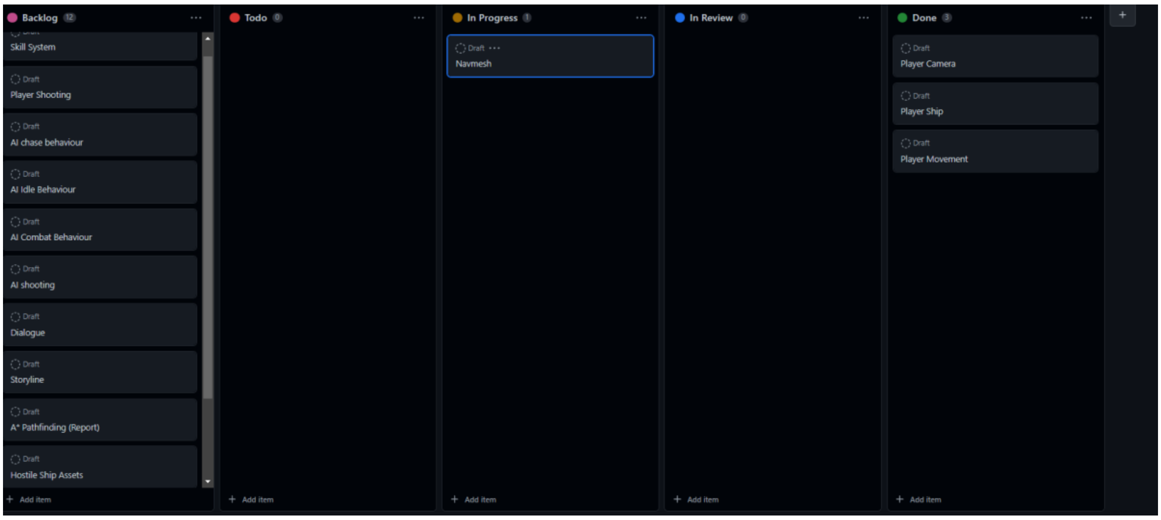
The player character was another core component to the game and so it was important that the controls were intuitive to use. The traditional format of using WASD as the input keys for movement felt the most appropriate and as such the player ship can move forwards by holding the W key, strafe left or right by holding the A or D key respectively or move backwards by holding the S key.

To compliment the controls, the environment needed to feel like it had an impact on the player. As such, the ability to move in any direction is impeded by needing to accelerate up to the max speed.

So that the player does not float indefinitely, drag was added in the form of a small force which felt like smaller inertia thrusters were dampening the speed of the ship so that it slowly floats to a stop.

Sprint Two

The second sprint was spent focusing on the pathfinding system that the non-player units would use to navigate the game world and interact with the player.



Backlog 12

Skill System

Draft
Player Shooting

Draft
AI chase behaviour

Draft
AI Idle Behaviour

Draft
AI Combat Behaviour

Draft
AI shooting

Draft
Dialogue

Draft
Storyline

Draft
A* Pathfinding (Report)

Draft
Hostile Ship Assets

Todo 0

In Progress 1

Draft
Navmesh

In Review 0

Done 3

Draft
Player Camera

Draft
Player Ship

Draft
Player Movement

Figure 7 Sprint two project board

NavMesh Pathfinding

The first attempt at creating a pathfinding system I intended on creating a NavMesh on which the non-player units, referred onwards as the AI, would traverse the game world using. I tried creating a plane on which the NavMesh could exist on but after a few days of tinkering I decided it would be better to create a dynamic system for the navigation and instead chose to go with an A* pathfinding system instead. The A* pathfinding script that my script is based off is from a youtuber known as Sebastian Lague. He has a ten-part tutorial on creating an A* pathfinding script. I have referenced certain parts of his works and created my own adjustments and alterations to his work to create the collection of scripts that make up the pathfinding.

A* Pathfinding

To create the A* pathfinding system, I first created two scripts that would create the nodes and the grid.

Nodes and Grids

The nodes script features a constructor for node creation as well as some variables to know the estimated cost of the journey so far and the journey left to go. There is also a variable and method to retrieve the index of the heap that the node is in. The size of the node can be adjusted to change the number of nodes in a grid so that if the computation time becomes too long it can be simplified by reducing the number of nodes to compute.

The grids script creates the grid by first taking a size from the inspector (for example 100) and locating the world bottom left. From there the script will then start populating the grid with nodes based on their size by calculating how many nodes will fit into the grid created by dividing the world size by the node size. While the nodes populate the grid, they check if they are in collision with anything and if so mark themselves so that they cannot be used for any pathfinding and instead are an object to go around. This is done by applying a mask to everything on the playable plane (at or around 0 in the Y axis) that will be in the way.

Heaps

For the heap structure, I opted to use a balanced binary tree heap. The purpose of the heap is to reduce the number of checks completed against different nodes that may not be of much use to finding an optimal path. To ensure that the nodes are in the right order, the heap will sort the nodes so that when a new node enters the heap, it will be checked against its parent and then if the parent is larger, will swap places with it, and repeating until there is no parent with a larger F cost (the total estimated cost of the journey). This method will also help to make sure that if a node is removed from the heap, then the most recently added node will be placed at the start of the heap and the comparison process is reversed, so it will check for the smaller of its two children and swap places with it. This process is repeated until there are no children smaller than itself.

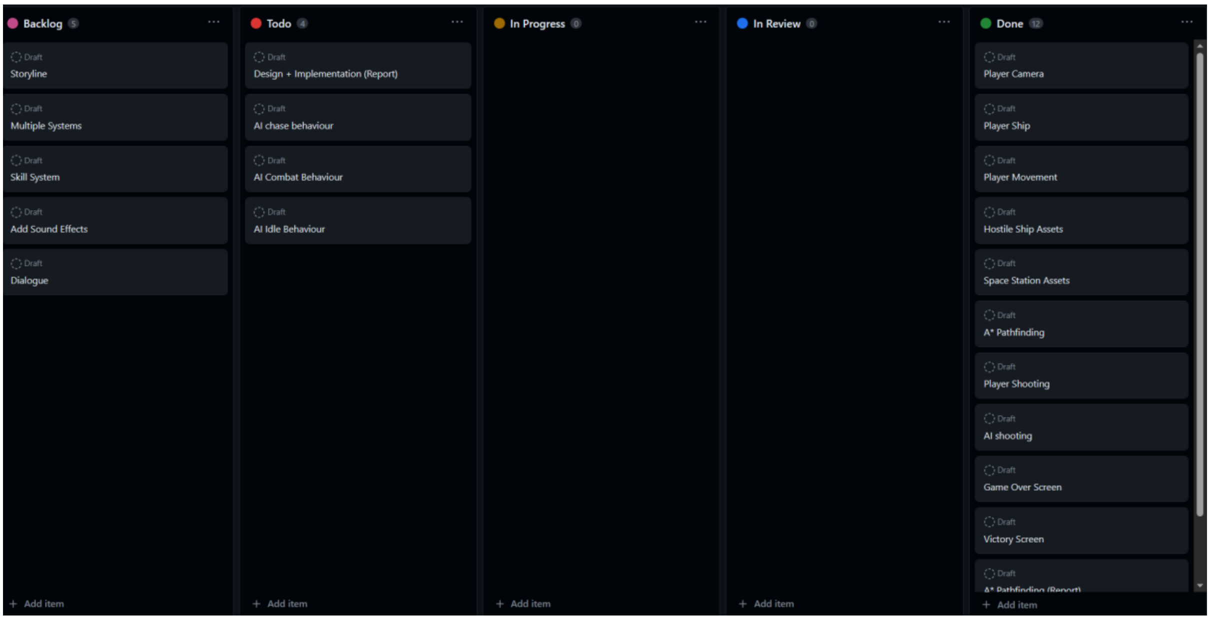
Sprint Three

The third sprint was spent focusing on my pathfinding system and making the code more readable. Each of the functions that make up pathfinding were updated to make the code more readable, with additional comments to aide understanding the use of each function and the naming reasoning behind certain methods / variables.

It was during this sprint that I decided to get a low poly asset pack from the unity asset store which contains many ship types, multiple station types, asteroids, props and much more. By adding this asset pack, I can reduce the amount of time developing my own assets. This is boon both because of the lack of experience I have in the area and so I could spend more time developing the code base for my project.

Sprint Four

During the fourth sprint, A lot of time was spent on simplifying level creation, creating gameplay elements, and implementing basic state behaviours for the hostile ships.



Backlog 5

Draft
Storyline

Draft
Multiple Systems

Draft
Skill System

Draft
Add Sound Effects

Draft
Dialogue

+ Add item

Todo 4

Draft
Design + Implementation (Report)

Draft
AI chase behaviour

Draft
AI Combat Behaviour

Draft
AI Idle Behaviour

+ Add item

In Progress 0

+ Add item

In Review 0

+ Add item

Done 12

Draft
Player Camera

Draft
Player Ship

Draft
Player Movement

Draft
Hostile Ship Assets

Draft
Space Station Assets

Draft
A* Pathfinding

Draft
Player Shooting

Draft
AI shooting

Draft
Game Over Screen

Draft
Victory Screen

Draft
A* Pathfinding (Report)

+ Add item

Figure 8 Sprint three project board

Prefabrication

This was done by ensuring that all the repeatedly used assets such as the player ship, the enemy ships and the bullets were prefabricated. By prefabricating certain elements of the game, I can now more easily add them into a scene without needing to add or remove components, change values such as scale, reload speeds, top speeds, acceleration, damage dealt and much more. After creating prefabrications, I started to create the scripts for bullets so that they could be instantiated, travel, collide with other objects and deal damage. I made the approach to split the bullets into two distinct variants, one which has a blue colouring, indicating to the player that these bullets are their own, and then another bullet which has a bright red colouring to signify to the player that the bullet came from a hostile ship. While the bullets damage does not differentiate between a friendly and a hostile, it can help the player identify which bullets they shot and which ones where shot at them. I created and added a damageable class to objects that can take damage such as the player ship and the AI controlled ships. This reduced some instances of code repetition by ensuring that the logic is the same and only written once in a single place. This means that I cannot change the value in two places and ensures consistency.

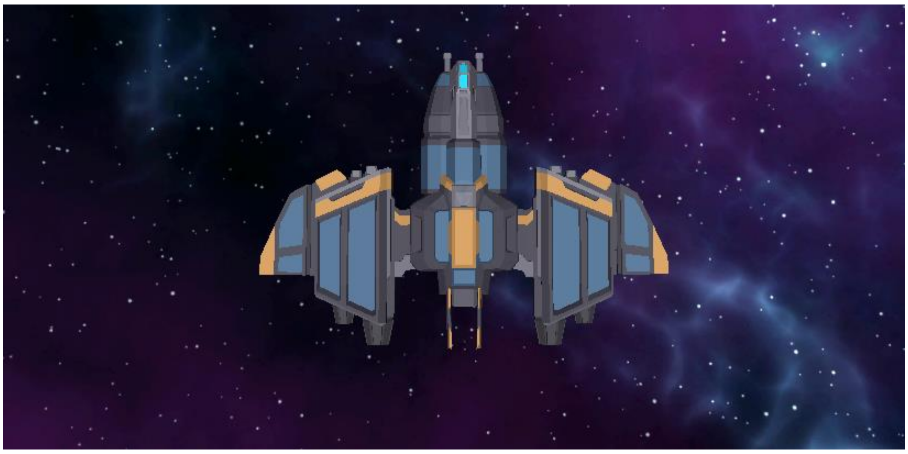


Figure 9 Player Ship

I realised that my bullets added a small impact force on the ships they were hitting other objects that had rigidbody components such as the hostile and player ships. To resolve this situation the bullet prefab was adjusted such that the function that detects when an impact occurs, the method "OnCollision" was changed to be an "OnTrigger" function. This meant that now the bullet did not transfer a force over to the collided object. I created a bullet origin game object for the bullets to leave from which would allow me as the developer decide at what position the bullet originated from. This is important as it will otherwise look as though the bullet leaves from the centre of the object firing it which is unintended and could cause issues with rigid bodies colliding. This was a simple addition by creating an empty game object as a child inside the prefab. Different ships have different fire origin points as the shapes of the ships are different and so it was important to take this precaution instead of adding an offset to bullet instantiation. This did however lead to some undesirable side effects such as bullets appearing to stand up. This was an easy fix as I could just rotate the fire origin such that the "up" direction was in the forward direction of the ship, so the bullets returned to their desired orientation. This also meant that the direction of travel in the script also needed to be changed from "transform.forward" to "transform.up".

Figure 10 Hostile Ship

AI Behaviour

The next important step was to create a state machine for the AI to use. The plan was to start simple and to include three main behaviours: patrolling, close distance, maintain distance. If I felt that more behaviours needed to be added after this, then it would be easy if I prepared for this. To create transitions between the states, a default state would need to be set, and for a ship defending an objecting, patrolling seemed like the perfect fit. The AI should start in the patrolling state and then when the player is a pre-determined distance from the defending point (a game object set somewhere in the play space), the behaviour will switch to closing distance and attempt to reduce the distance between the player to begin engaging them. If the ships were then too close, the behaviour would swap to maintaining the distance, where a negative thrust will be applied to try increasing the distance from the player. If while inside either close distance or maintain distance and the player ship left the range of the defence point, then the behaviour will return to patrolling where they will travel to the next checkpoint along their patrol point.

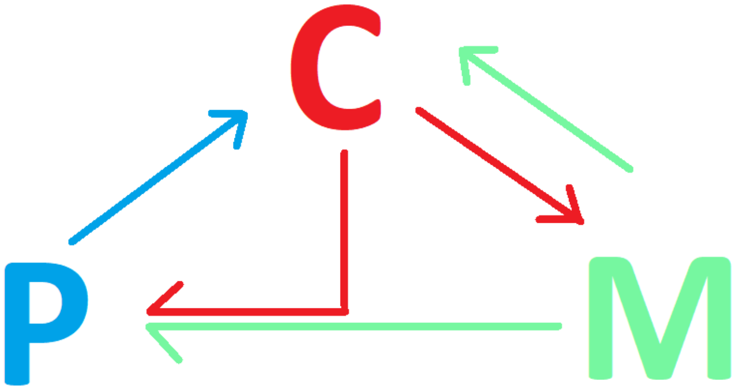


Figure 11 AI state behaviours and their transitions

Patrolling

The patrolling behaviour is the default behaviour that the AI will be set to upon spawning. The state can be reached while in the close distance or maintain distance behaviours if the player passes the disengage radius. The AI ship will then look toward the waypoint and travel in that direction until they reach the threshold that is set to consider them within the checkpoint. Once the ship has reached the checkpoint, the pointer of the array of checkpoints increments, and the ship will begin to travel toward the next waypoint. Once the pointer is at the length of the array, the pointer is set back to the first index. The AI at this stage does not path around objects and just uses a simple look and move procedure, but this is developed in the next sprint to include the pathfinding algorithm to path around obstacles. The diagram below shows the desired outcome of the patrol behaviour.

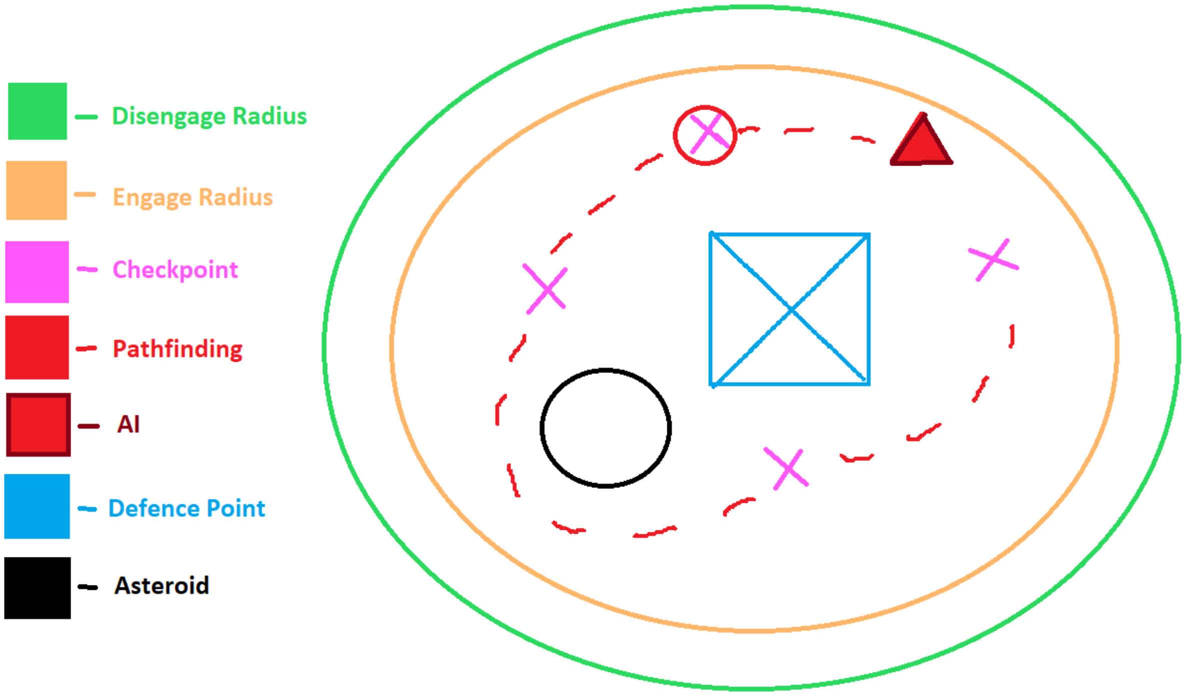


Figure 12 AI patrol behaviour

Close Distance

The transition into closing the distance between the AI and the player occurs if the player is within the engagement radius of the defence point. It is important to note that the reason the engagement radius and disengagement radius are two separate values as opposed to just sharing the same is that the player could exploit the behaviour by sitting just outside of the range and "poke" at hostiles without them being able to fire back as they would be attempting to return to their patrol route. The Close distance behaviour makes the AI look toward the player and begin applying a thrust to reduce the distance between the two. It is during this behaviour that the AI will also perform several functions to determine whether they are able to shoot at the player. This includes checking if the ability to fire is ready (for example it has been at least 2 seconds since the last shot was fired), checking whether the player is within the acceptable angle to begin attacking (as demonstrated in the diagram below), and that the distance between the player and itself are within a determined range. At this point the AI will attempt to fire a shot at the player. If the player decides to pass the disengage radius from this point, the AI state will return to patrolling. If while closing the distance to the player, the gap between the two surpasses a threshold, the state will swap over to maintaining distance.

Maintain Distance

While the AI is attempting to maintain distance, it will continue to look toward the player and, provided that the requirements for shooting are met, will continue to shoot at the player. This behaviour mimics many of the close distance behaviours with the only main change being that the thrust applied to reach the player is inversed, so that the ship-to-ship distance increases instead of decreasing.

Firing

The AI will perform many checks to ensure that it can fire at a player. The first thing the AI checks is whether the player is within the maximum distance to fire at. If this is true, the next step is to calculate the distance between the fire origin (where the bullets come from) and centre mass of the player ship (also known as a vector 3 and shown as a dark blue line below). The next step is to calculate the angle between the front of the ship (the orange line) and the player ship position. The pink angle represents the returned angle. From this point, the AI checks whether the angle returned is within the max firing angle threshold (max angle and shown in green). The max angle is unsigned and so the variable is half of the value desired. Theoretically, it could be adjusted so that the value input is divided by 2, meaning the value entered is the angle in front of the ship, and not the angle to check in each direction from the front. If these conditions are met and the ship has reloaded (shoot cooldown variable is less than or equal to 0) then a bullet is fired. All of this happens while the AI is trying to look toward the player ship at the same time, to keep it as close to in front as possible.

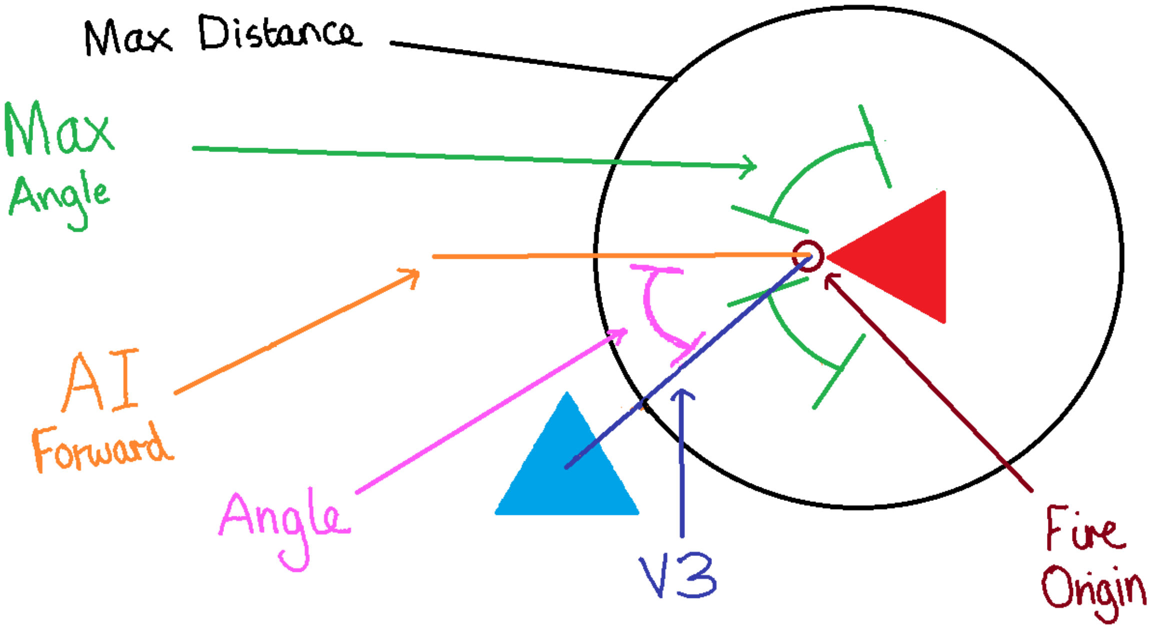
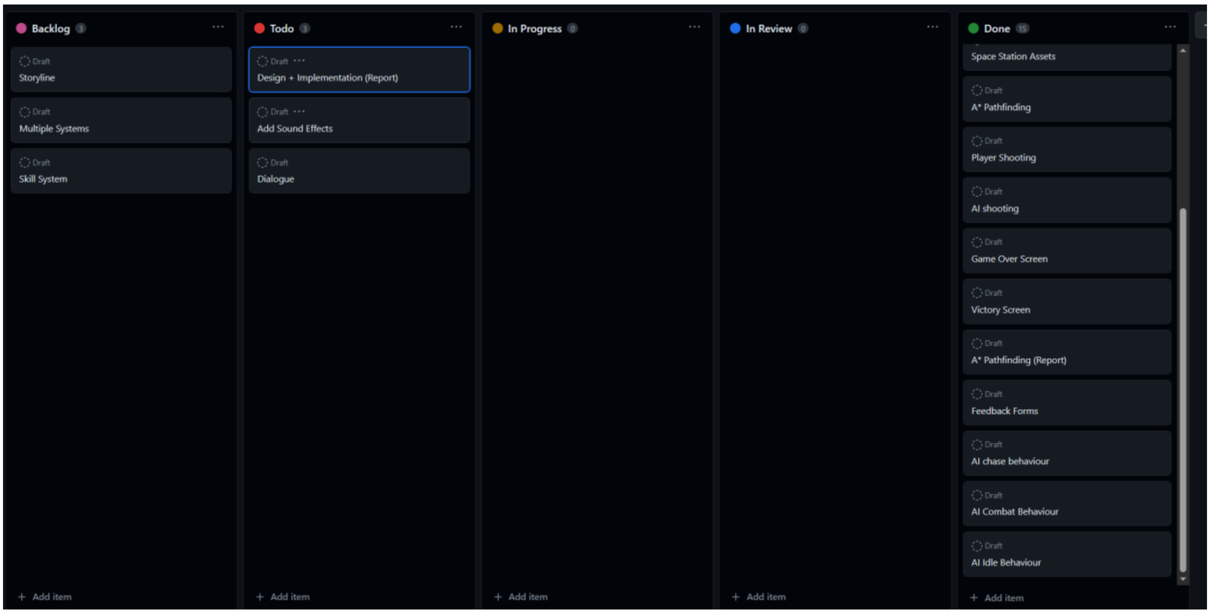


Figure 13 AI engaging behaviour.

Sprint Five

During the fifth sprint, many of the final components of the demo where added. This included sound effects, and a dialogue box to introduce the player and the remainder of the time devoted to report writing. Seen below is the introductory dialogue that the player views as they load up the application. It has yellow arrows that indicate to the player that it must be interacted with to be cleared.



Backlog 3

Draft
Storyline

Draft
Multiple Systems

Draft
Skill System

+ Add item

Todo 3

Draft
Design + Implementation (Report)

Draft
Add Sound Effects

Draft
Dialogue

+ Add item

In Progress 0

+ Add item

In Review 0

+ Add item

Done 15

Space Station Assets

Draft
A* Pathfinding

Draft
Player Shooting

Draft
AI shooting

Draft
Game Over Screen

Draft
Victory Screen

Draft
A* Pathfinding (Report)

Draft
Feedback Forms

Draft
AI chase behaviour

Draft
AI Combat Behaviour

Draft
AI Idle Behaviour

+ Add item

Figure 14 Sprint five project board

Adding the sound effects consisted of first adding a sound component to anything that needed to create noise. This was the bullet, and all ships. At first, I used some sound effects created ever so kindly by my girlfriend, shown in the figure below, she first made a “pew” sound for the firing of the guns, and another noise for the explosion. After importing them into unity and playing around with the audio settings, the explosion sound was often too quiet and so I used a media creation tool known as BFXR to create sound effects to compare. I personally enjoy the sounds that my girlfriend created but the sound effects I managed to create using the software sounded closer to the true sound. While ultimately the two sets of sound effects can be used interchangeably, for the use of this demo, I have opted to use my girlfriend’s shooting sound, and the software’s explosion effect. This met the perfect middle ground of loud noise, and a personal touch for me.

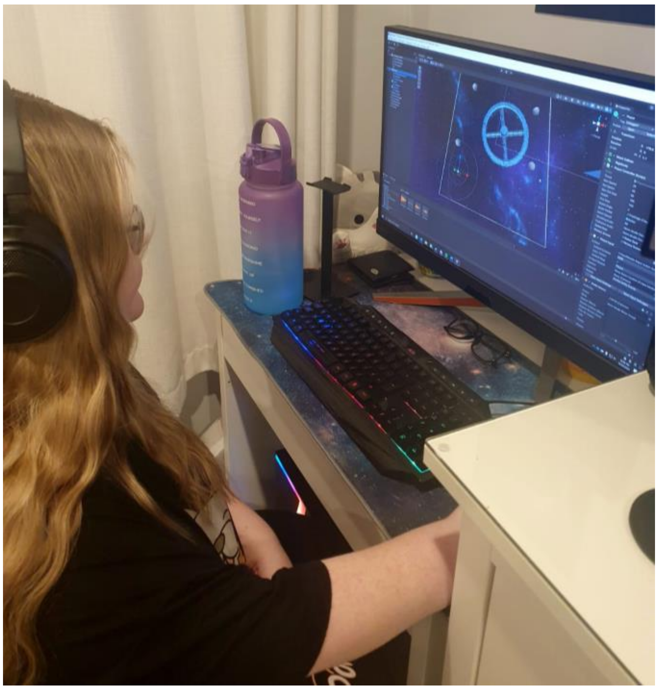
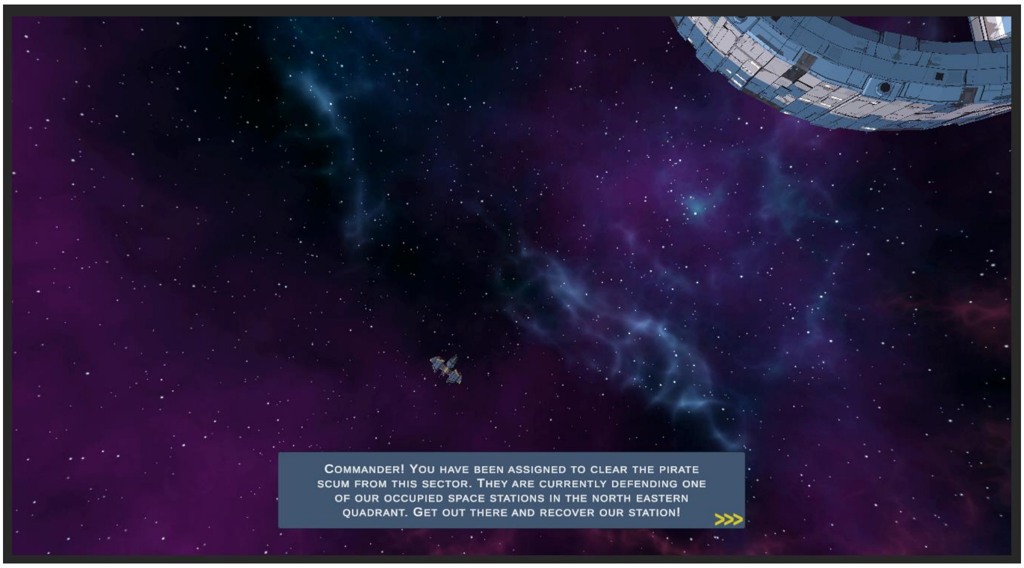


Figure 15 My girlfriend listening back to the audio files she recorded.

Final Demo Build

Once I had created my demo, I progressed through the level and tested many of the features. In figure 16, I demonstrate the initial starting position and dialogue box that appears for the player. There are also yellow arrows to show the player that it is waiting for input to progress.



COMMANDER! YOU HAVE BEEN ASSIGNED TO CLEAR THE PIRATE SCUM FROM THIS SECTOR. THEY ARE CURRENTLY DEFENDING ONE OF OUR OCCUPIED SPACE STATIONS IN THE NORTH EASTERN QUADRANT. GET OUT THERE AND RECOVER OUR STATION!

Figure 16 Introductory dialogue box

In figure 17, I showcase some of the other assets in my game. In the upper left corner, an asteroid is seen and the player ship can be seen shooting bullets toward the hostile space ship, which is currently patrolling above the station.

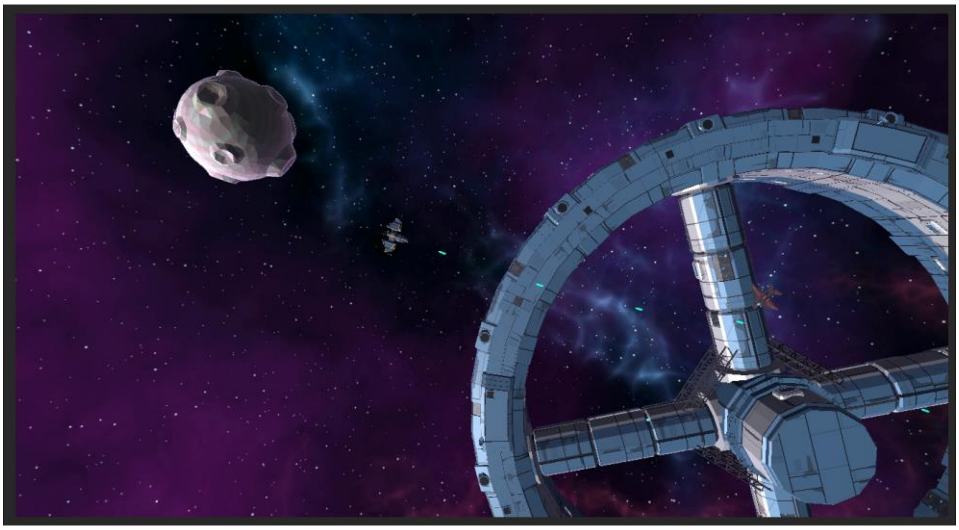


Figure 17 player shooting bullets toward hostile.

As seen in figure 18, the two hostile spaceships that are present in the demo level are engaging the player, closing their distance as they fire shots toward them. Both ships are trying to chase the player as they back away from the hostiles, dodging bullets by strafing to the right at the same time.

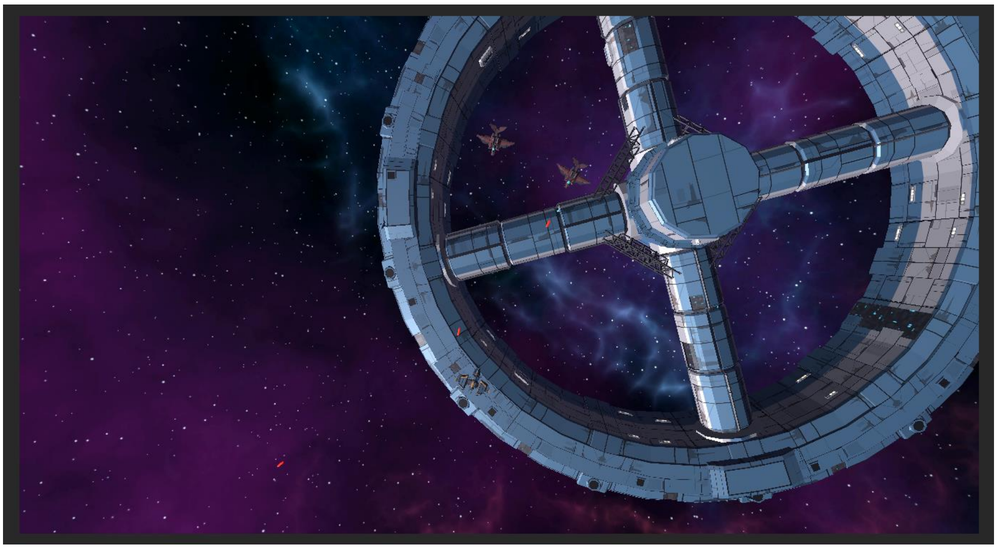
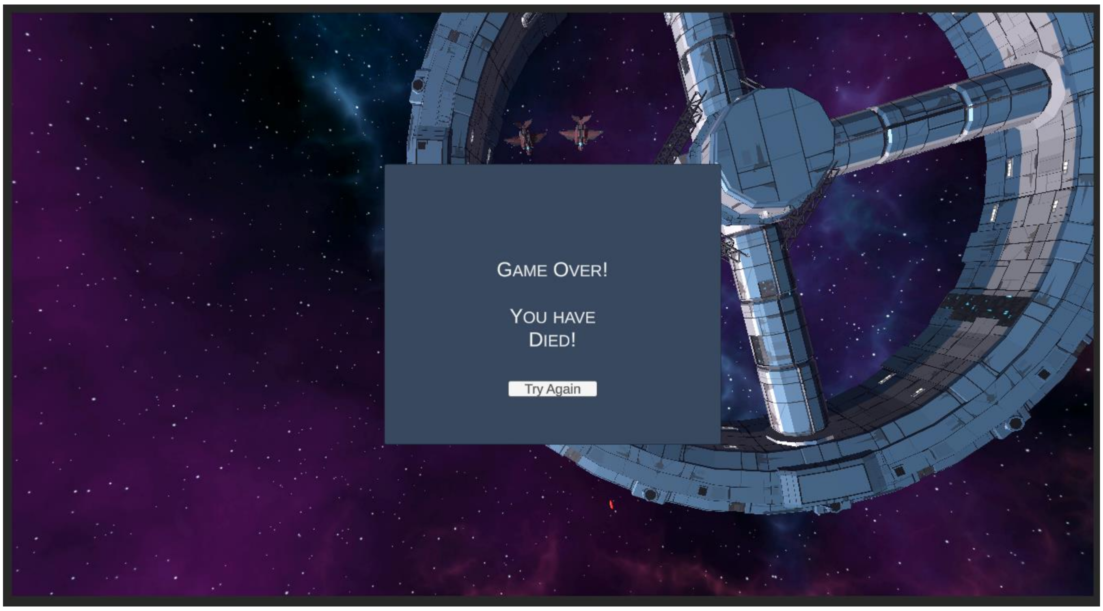


Figure 18 hostiles engaging the player.

Figure 19 shows the game over screen once the player's hit points (which as of the current build cannot be seen) have reached zero and the player is considered dead. Upon pressing the try again button, the level is reloaded, showing again the dialogue, and allowing the player another attempt.

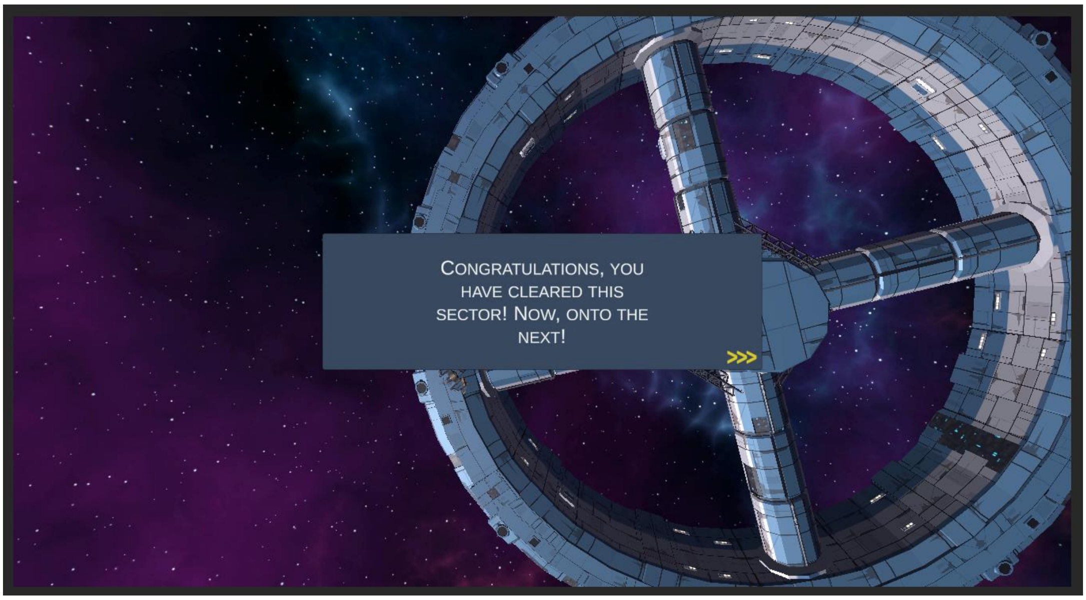


Game Over!

You have Died!

Figure 19 game over screen.

Once the player has defeated the pirates, a dialogue box like the one showcased in figure 20 will display. At this moment in time there is no further action that can be done but the same design process was applied to the sector cleared dialogue as the initial dialogue, suggesting the player clicks the dialogue to progress to the next sector.



Congratulations, you have cleared this sector! Now, onto the next!

Figure 20 sector cleared.

As the development time for the project finished, I took one last screenshot of the project board (figure 21) which was originally populated with tasks to complete the mandatory objectives, and the only items left in the backlog are a couple of extension objectives that will be added at a later date.

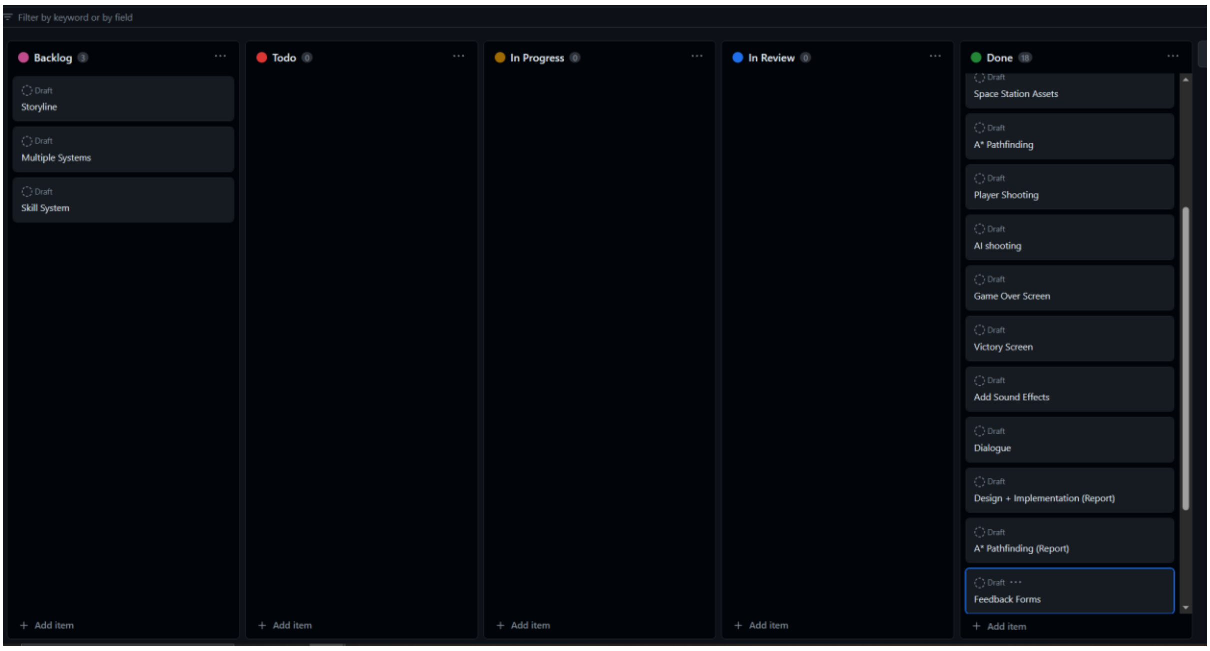


Figure 21 End of development project board

User Testing

When creating a project designed to entertain, collecting some feedback from the people who will be using your project is useful.

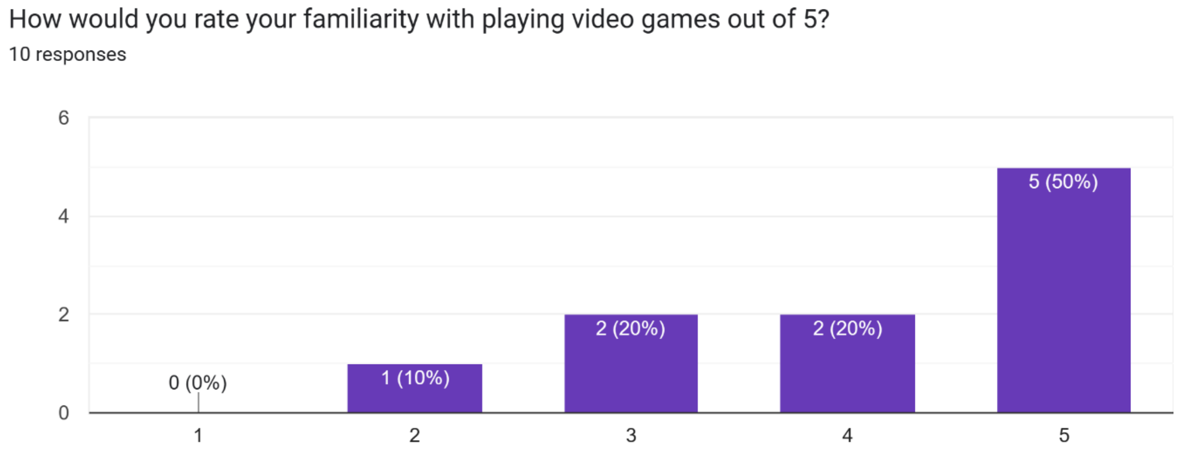
> *"Getting real players interacting with your prototypes early and regularly in development can be uncomfortable. Yet it ensures teams are never out-of-touch with where design implementation is falling short, or headed for a design dead-end."* (Player Research, 2023)

Player Research's approach to playtesting comes down to four core principles:

- Always involve a member of the public playing a prototype
- Typically, 6-12 play testers playing 90 minutes at a time.
- Play testers are invited to "play as you normally would", sometimes thinking out loud while playing.
- A researcher moderates the session, collecting player behaviour data.

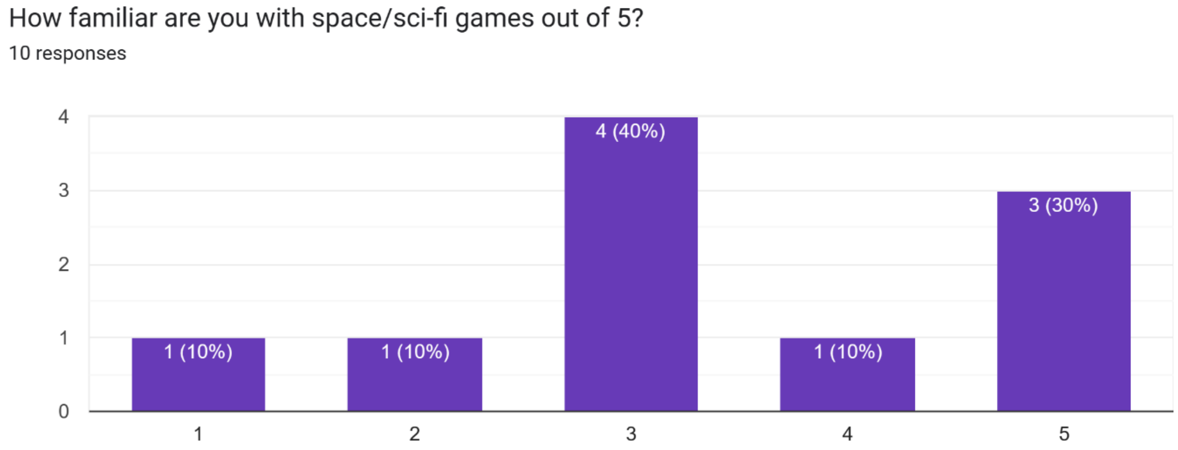
The information collected indicates what has gone well, what has not gone so well, some things that could use some improvement and an assessment of the reviewer's ability and relevance to the project. For example, if a user has not played board games before, they will have a different experience to those who have had a lot of experience. All of this can be collected using a variety of open and closed questions. The collection of closed questions gives useful quantitative data that is measurable. This type of data is best suited to finding averages and for questions where you want to offer a range of answers rather than any possible answer. The types of questions I asked for my closed questions included the reviewer's experience with games, with sci-fi games, reviewing/playtesting a game, accessibility, and pre-set answers for the difficulty that best describes the demo that the reviewers played. Open Questions are best used for getting a more descriptive response, where the visualisation of the data is not important. There are a few questions that are open on the review form such as asking for the most enjoyable feature, and what feature would they like to see most in the complete version of the game.

Of the people who played my game, 70% said they were comfortable playing games to a reasonable level. This indicates that any results from the review will be from people who have experience playing games. This is both a good thing and a bad thing as it means that while more critical points can be plucked from the demo that they played, some more basic aspects such as the controls and difficulty may not be properly represented. It is also important to note that only 20% of the testers were not very familiar with the genre of the game. This could suggest that the data collected may be more accurate and is being compared against similar games in the genre.


How would you rate your familiarity with playing video games out of 5?

10 responses

Figure 22


How familiar are you with space/sci-fi games out of 5?

10 responses

Figure 23

Of all the participants, 30% had not reviewed or play tested a game before. While this could influence the review, the participants will have some experience of critical thinking and have had experience in reviewing other products, even if they have not formally reviewed or play tested a product.

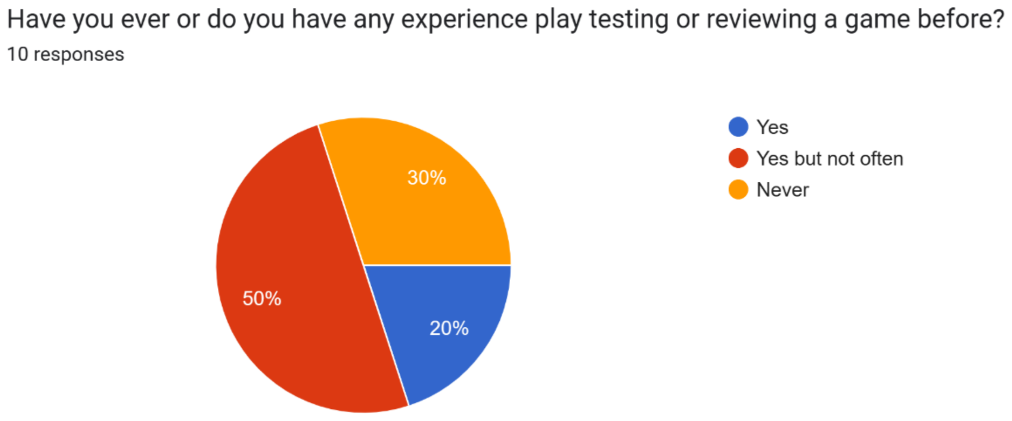


Figure 24

The overall usability of the project could be considered within acceptable bounds as 90% of participants scored at least 3 or higher.

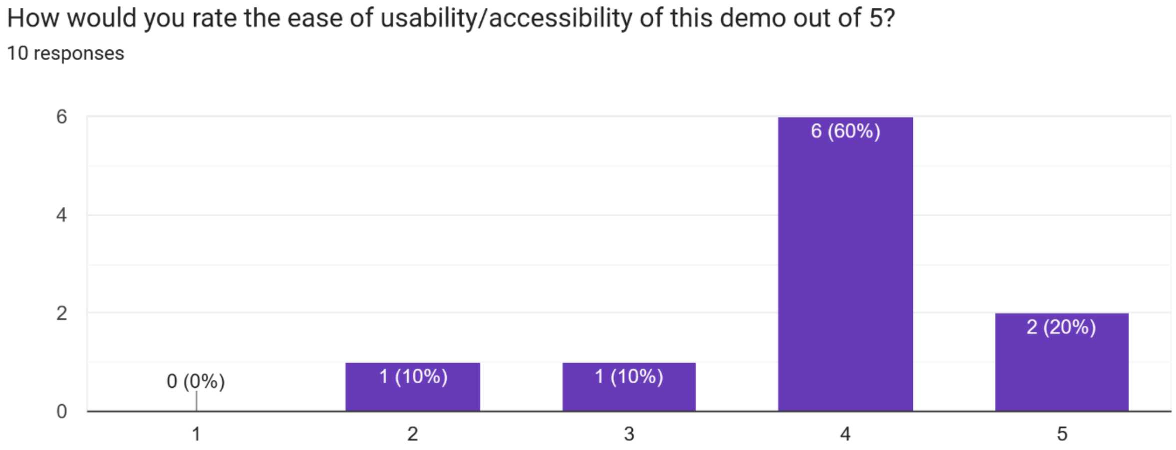
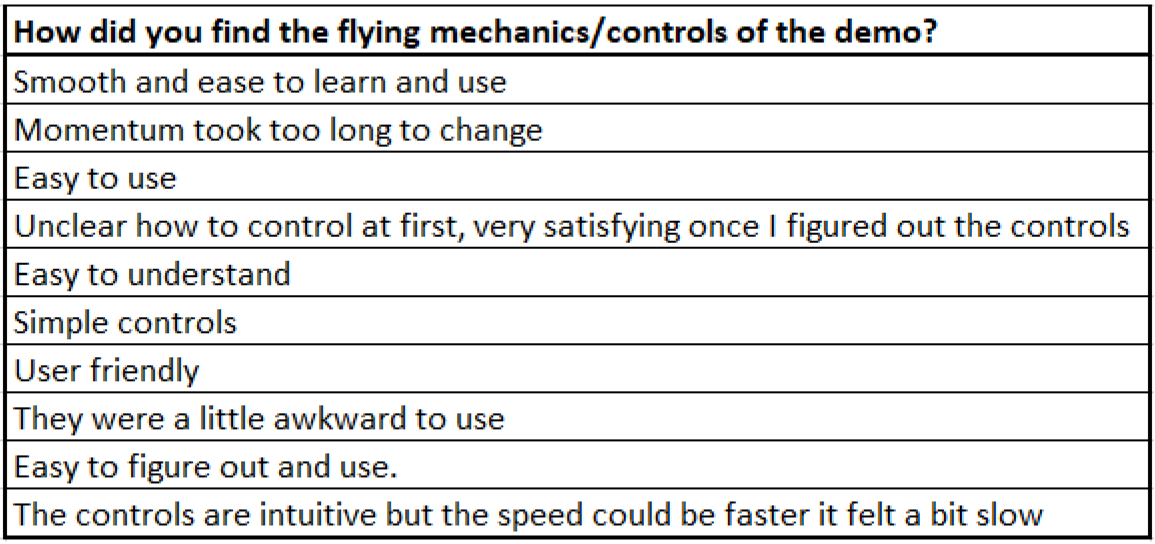


Figure 25

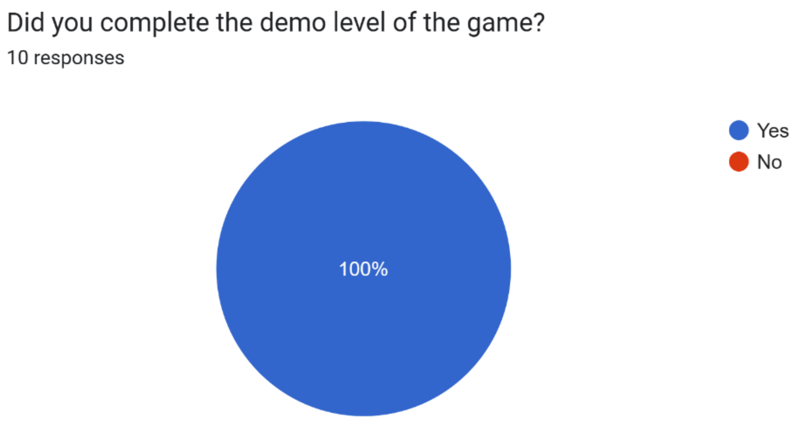
Results from the flying mechanics and controls show that only 10% of participants found the controls awkward to use. The rest of the participants found the controls easy to understand/use. There is some mention of the acceleration rate being too slow. This is an easy adjustment of changing two variables for both the player and the AI ships. This would also need further user evaluation to find the correct range of speeds.



How did you find the flying mechanics/controls of the demo?
Smooth and ease to learn and use
Momentum took too long to change
Easy to use
Unclear how to control at first, very satisfying once I figured out the controls
Easy to understand
Simple controls
User friendly
They were a little awkward to use
Easy to figure out and use.
The controls are intuitive but the speed could be faster it felt a bit slow

Table 1

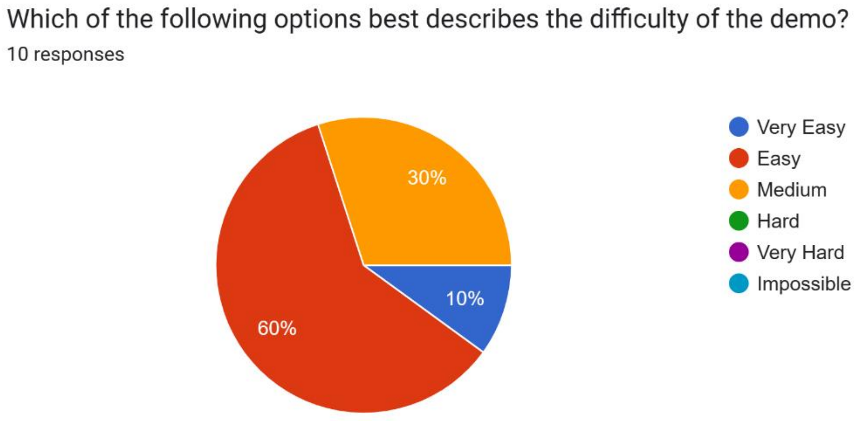
Data regarding the overall difficulty of the level suggests that the demo level was not of a high enough difficulty. While 30% of the participants state that the difficulty was of a medium level, 100% of the total participants completed the demo with only 30% dying at least once, which could suggest that they were simply testing what would happen should they die. Overall, the consensus was that the game would need to be harder than the current situation. This could be remedied by using more enemies per encounter, or to increase the speed of the game so that the reaction time would need to be faster.



Did you complete the demo level of the game?

10 responses

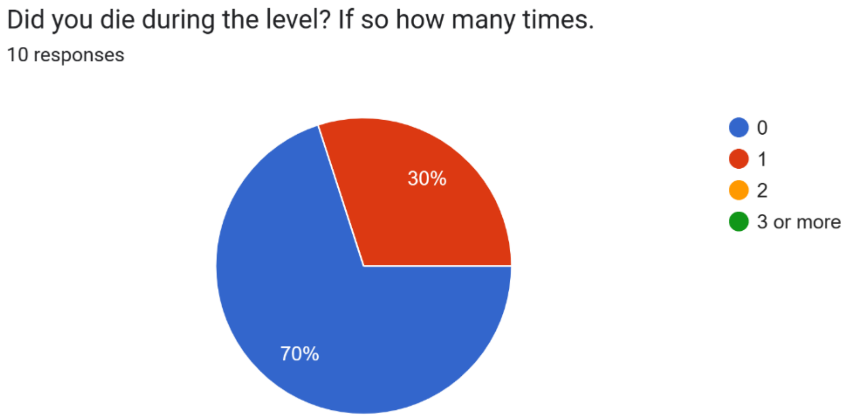
Figure 26



Which of the following options best describes the difficulty of the demo?

10 responses

Figure 27



Did you die during the level? If so how many times.

10 responses

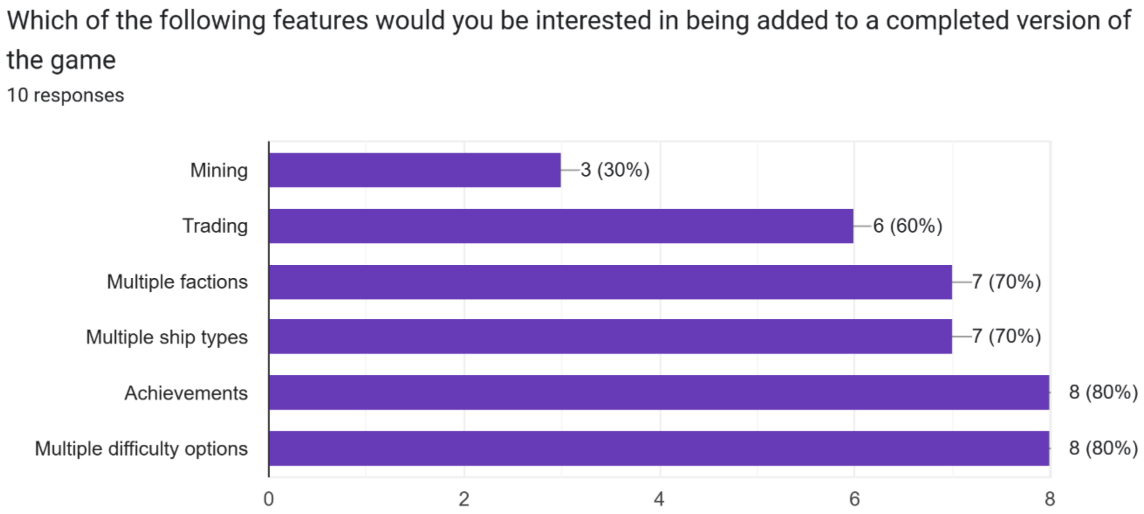
Figure 28

I can see that from the positive experience answers, the combat felt satisfying, and the game was visually appealing.

What did you like/find most enjoyable about the game?
Shooting the enemies
The enemy movement was good
The graphics
The satisfying movement for flying the ship
Flying around space
Killing the enemy pirates
Patrolling for pirate ships and defending space stations
The ai looked smart and were accurate at shooting
Completing an objective, and cute sound effects
I really enjoyed the look of the game very aesthetically pleasing

Table 2

The participants then commented on what features they would be interested seing from a list of possible options. This data suggests that mining was a feature that was not very desirable but that almost all participants (70% or higher) would like to see multiple factions to be involved in the game, multiple ship types, achievements, and multiple difficulty options. Creating multiple factions could introduce a new dynamic to the game where there are factions that the player can view as an ally or two factions vying for power who also see the player as a hostile. These different situations lead the way into creating a world that feels like the play has a bigger impact and has more life in it. This data also shows that mining was a feature that less than half of the half of the participants felt should be added to a complete version of the game. Mining would give more goods to trade with stations, however this could also be adjusted so that the resources that would normally be obtained by mining such as ore and ice can be purchased from stations to be sold again.



Which of the following features would you be interested in being added to a completed version of the game

10 responses

Figure 29

The survey showed that 100% of the participants would be willing to complete the game at least more than once. All the participants would also be willing to play a full version of the game when released. The acceptable price range for the game was between £3 and £10 with an average acceptable price of £6. This price would assume a full complete game with multiple levels and additional features.

Evaluation

Overall, my project was a success. I completed all the mandatory objectives, constructed a demo level and the prototype of the game I have produced is in a completed state and all the pieces are together to easily produce a fully functioning version of the game. I have a prefab of the player and the hostile ships, I have a full asset pack including environmental objects including but not limited to stations, asteroids and ships, there are sound effects in the game and are included with the prefab of the player and AI. The act of creating a new level is now as easy as dragging in a player prefab from the assets folder, some hostile ships, creating a defence point and some patrol checkpoints, assigning those to the hostile ships and placing environmental objects. The dialogue of the game can be added once more levels exist.

The mandatory objectives were designed to provide a point where all the pieces of a game were constructed. The game successfully runs on Windows 10 as of the time of the report being written. The player can move in any direction by using the W A S and D keys. The game is viewed by a top-down camera which glides toward the mouse and as such allows more of the players "front" view to be seen by pushing the player to the edge. The player can fire projectiles with the left mouse button. The game features a hostile ship prefab and as such as many enemies can be created with very small amounts of effort. The asset pack that I used for the game features station assets and asteroid assets that can be used. While health is not visible at any point, there is an adjustable hit point value for both player and hostile ships assigned in the prefab. While the game does not have a winnable state as such, there is a dialogue box that will tell the player that they have cleared the sector. This serves as an announcement to the player that they have completed the level. The game also features a dialogue box that introduces the player to the world they are in and gives enough context to understand what is happening. This storyline would be expanded on as the player progressed through each level. The game features a sound effect for when a ship shoots a projectile and another sound effect for when a ship's health pool reaches 0 and is destroyed.

I intend on further developing the project upon submission to complete many of the extension objectives and will break down how they will be implemented if I had more time during the project to do. The current version of the game does not feature any save points, so progress is lost upon the application closing. Unity features a save system in the game which, with some additional user interface elements, can add a main menu with saving and loading. There are a small number of station types available I the asset pack I obtained for the project; it does not meet the 5 types I mentioned in my extension objectives. While the term "ship types" is vague, the asset pack I have contains many different ship classes (such as fighters, cruisers, frigates etc) and many varieties of those different classes. Currently only two fighter varieties are in use, one model is the player, and a second model (also a fighter variant) is used for the pirate ships. As stated in the extension objectives, a difficulty option would be added if I had more time. Through the user survey, the necessity for this feature has been explored. This same approach applies to adding achievements to the game to give the player a sense of accomplishment. There could achievements for a variety of tasks such as one for completing the game on each difficulty level, defeating a certain number of enemies within a predetermined time, not taking any damage in a sector and much more. Accommodating for people who have hearing, visual and motor disabilities would increase the total number of players able to enjoy the game as comfortably as possible. Without further development and a large time investment, it is difficult to add these adjustments. I would need to have a menu system where the player can set the correct adjustment to aid them. This is something that would be done closer to the full version of the game once all the other features are added. There are other systems that, from the user testing, could be put in place to improve the game and would be a worthwhile addition, these include multiple factions that the players could interact with. Introducing multiple factions would then lead to an honour or reputation system that would be implemented to punish the player for doing actions that would be deemed unacceptable by the target faction, such as shooting down their ships. This could have further effects on the player by making them a hostile faction, increasing the number of threats the player faces. This would be a nice addition to give the player some control over their world and increase the chance that they talk with people about the game and their experience, which could increase the reach of the game. Multiple systems would be included in the full version of the game, along with the narrative storyline. This would bring the total number of systems up, and with that a scaling difficulty where the health of the enemies, their damage and more could be adjusted to make it harder for the player and add a sense of progression. This would tie in nicely with a skill system where the player can improve their own stats at the same time such as their top speed, acceleration, rate of fire, the damage their weapons do and much more. With increased difficulty there is the necessity to add a way for players to restore their hit points after or between battles. This could come in the form of a consumable or the ability to repair at a friendly station, possibly with some form of currency.

References

Audacity. (2022, 10 August). *Audacity*. Retrieved from Audacity: https://www.audacityteam.org/

Elo Entertainment Inc. (2022, November 14). *speedrun.com*. Retrieved from Speedrun.com: www.speedrun.com

Koster, R. (2013). *a Theory of Fun for Game Design.* O'Reilly Media Inc.

Lague, S. (2023, January 12). *A* Pathfinding*. Retrieved from Youtube: https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXl0cq5Umv3pMC9SPnKjfp9eGW

Lavelle, S. (2023, April 04). *BFXR*. Retrieved from BFXR: https://www.bfxr.net/

Mojang. (2022, November 15). *Minecraft*. Retrieved from Minecraft: www.minecraft.net

Player Research. (2023, March 18). *Getting More From Usability Testing Your Game*. Retrieved from Medium: https://playerresearch.medium.com/getting-more-from-usability-testing-your-game-c87662065531#:~:text=U%20sability%20testing%20is%20the%20most%20powerful%20tool,falling%20short%2C%20or%20headed%20for%20a%20design%20dead-end.

ROCKFISH Games. (2022, September 19). *Games*. Retrieved from Rockfish games: https://rockfishgames.com/

Sony Interactive Entertainment. (2023, 04 25). *Ghost of Tsushima*. Retrieved from Playstation: https://www.playstation.com/en-gb/games/ghost-of-tsushima/

Still Alive Studios. (2022, September 19). *Drone Swarm*. Retrieved from Drone Swarm: https://www.droneswarmgame.com/

Synty Studios. (2023, March 08). *POLYGON Sci-Fi Space*. Retrieved from Unity Asset Store: https://assetstore.unity.com/packages/3d/environments/sci-fi/polygon-sci-fi-space-low-poly-3d-art-by-synty-138857

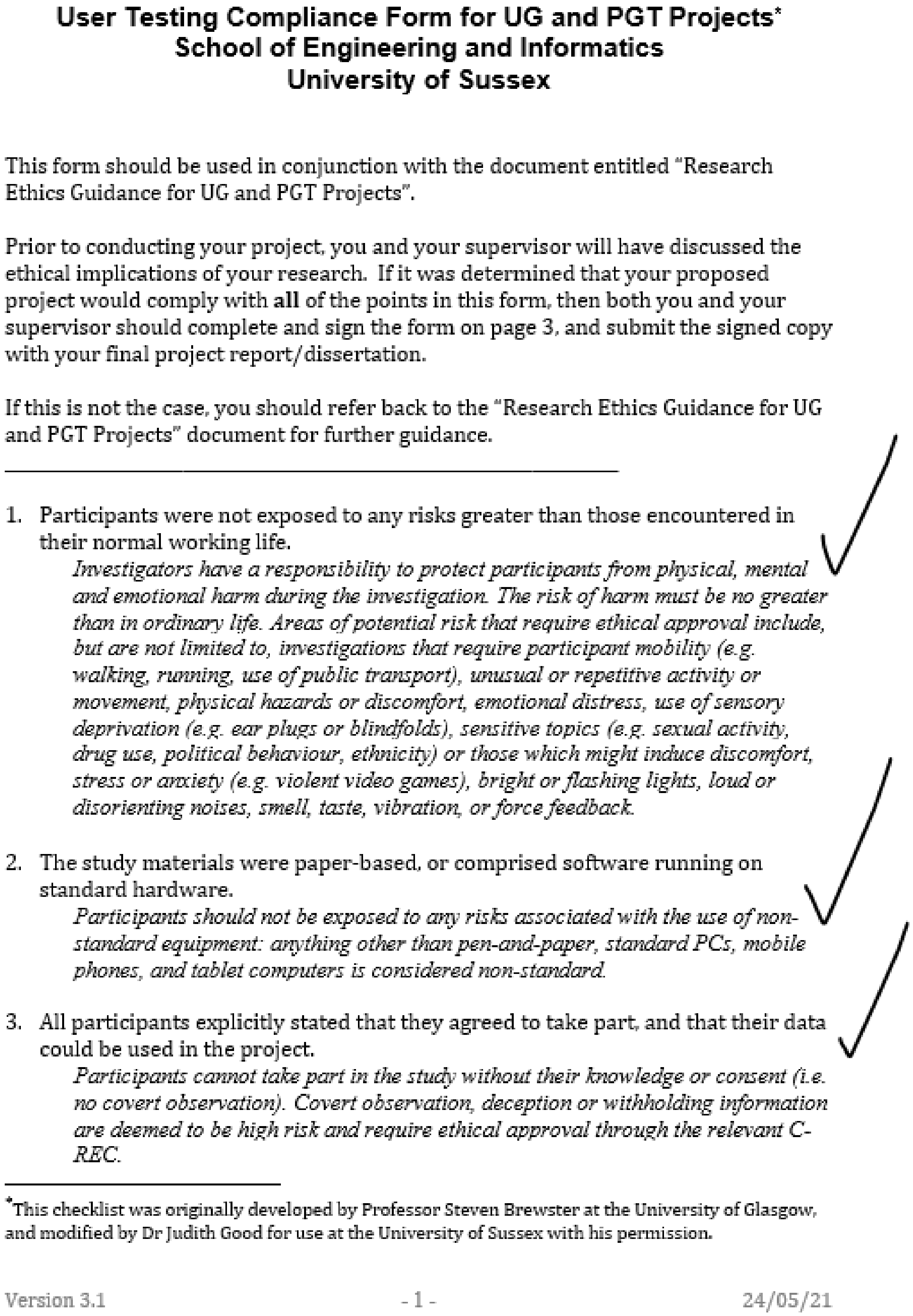
The Blender Foundation. (2022, 11 7). *Blender*. Retrieved from Blender: https://www.blender.org/

Unity. (2022, 11 7). *Unity 3D*. Retrieved from Unity 3D: https://unity.com/

Walternate Realities. (2022, September 19). *Cosmoteer*. Retrieved from Cosmoteer: https://cosmoteer.net/

Appendices

1. User Testing Compliance Form



User Testing Compliance Form for UG and PGT Projects*
School of Engineering and Informatics
University of Sussex

This form should be used in conjunction with the document entitled "Research Ethics Guidance for UG and PGT Projects".

Prior to conducting your project, you and your supervisor will have discussed the ethical implications of your research. If it was determined that your proposed project would comply with **all** of the points in this form, then both you and your supervisor should complete and sign the form on page 3, and submit the signed copy with your final project report/dissertation.

If this is not the case, you should refer back to the "Research Ethics Guidance for UG and PGT Projects" document for further guidance.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life. ✓

   Investigators have a responsibility to protect participants from physical, mental and emotional harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, physical hazards or discomfort, emotional distress, use of sensory deprivation (e.g. ear plugs or blindfolds), sensitive topics (e.g. sexual activity, drug use, political behaviour, ethnicity) or those which might induce discomfort, stress or anxiety (e.g. violent video games), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback.

2. The study materials were paper-based, or comprised software running on standard hardware. ✓

   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and tablet computers is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project. ✓

   Participants cannot take part in the study without their knowledge or consent (i.e. no covert observation). Covert observation, deception or withholding information are deemed to be high risk and require ethical approval through the relevant C-REC.

*This checklist was originally developed by Professor Steven Brewster at the University of Glasgow, and modified by Dr Judith Good for use at the University of Sussex with his permission.

If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, the data is to be published or there are future secondary uses of the data), then it will be necessary to obtain signed consent from each participant. Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script (see Appendix 1).

4. No incentives were offered to the participants. ✓
   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle. People volunteering to participate in research may be compensated financially e.g. for reasonable travel expenses. Payments made to individuals must not be so large as to induce individuals to risk harm beyond that which they would usually undertake.

5. No information about the evaluation or materials was intentionally withheld from the participants. ✓
   Withholding information from participants or misleading them is unacceptable without justifiable reasons for doing so. Any projects requiring deception (for example, only telling participants of the true purpose of the study afterwards so as not to influence their behaviour) are deemed high risk and require approval from the relevant C-REC.

6. No participant was under the age of 18. ✓
   Any studies involving children or young people are deemed to be high risk and require ethical approval through the relevant C-REC.

7. No participant had a disability or impairment that may have limited their understanding or communication or capacity to consent. ✓
   Projects involving participants with disabilities are deemed to be high risk and require ethical approval from the relevant C-REC.

8. Neither I nor my supervisor are in a position of authority or influence over any of the participants. ✓
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any study.

9. All participants were informed that they could withdraw at any time. ✓
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script (see Appendix 1).

10. All participants have been informed of my contact details, and the contact details of my supervisor. ✓
    All participants must be able to contact the investigator and/or the supervisor after the investigation. They should be given contact details for both student and supervisor as part of the debriefing.

11. The evaluation was described in detail with all of the participants at the beginning of the session, and participants were fully debriefed at the end of the session. All participants were given the opportunity to ask questions at both the beginning and end of the session. ✓

    Participants must be provided with sufficient information prior to starting the session, and in the debriefing, to enable them to understand the nature of the investigation.

12. All the data collected from the participants is stored securely, and in an anonymous form. ✓

    All participant data (hard-copy and soft-copy) should be stored securely (i.e. locked filing cabinets for hard copy, password protected computer for electronic data), and in an anonymised form.

Project title: Libra Imperius

Student's Name: Josh Withall

Student's Registration Number: 21902013

Student's Signature: ____

Date: 10/11/2022

Supervisor's Name: Dr Steve Huckle____________________________________

Supervisor's Signature: S Huckle ______________________________

Date: 10/11/2022____