|  |
| --- |
| University Of The West Of Scotland |
| The Mayhem Pits |
| Design Document |

|  |
| --- |
| Alasdair Hendry B00330023  11-28-2018 |

# Overview

The Mayhem Pits is a multiplayer game that can be experienced in virtual reality, using the Oculus Rift, or in a standard 2D display mode. These display modes are interchangeable, meaning that a player using an Oculus Rift is able to play online with a player using a standard display.

The Mayhem Pits is an arena-based, robot war game. The player is able to unlock and upgrade their robot whilst competing in the arena to earn more coins.

# Core Objective

As the game is multiplayer-based, the core objective is centred around winning as many online matches as possible. To do this, the player can choose to join either a two player, three player or four player game.

Once the match has started, each player then proceeds to destroy each other by using their weapons and the surrounding environment. The last player standing is declared the tournament winner.

# Features Implemented

The Mayhem Pits has a wealth of features that are available to the users. Each feature presents the player with new things to do or experience.

## My Robot

To begin with, the player has access to the MyRobot feature, in which they build, design and upgrade their robot using coins gained by winning tournaments. The MyRobot feature is only available to players who have created an account and are actively signed in, as their data is stored in the cloud. Customisations available in MyRobot include;

* Customising the body of the robot
* Customising the weapon used on the robot
* Customising a spring-based emblem that is attached to the robot
* Customising a skin that is wrapped around the robot

## Replay System

The Replay System is another prominent feature that has been implemented in The Mayhem Pits. Whilst this is not something that the player can interact with directly, it does provide another degree of immersion.

During moments of interest such as high-velocity impacts or eliminations, a replay of the previous 10-15 seconds of action is display on billboards that are placed through the arena. These replays are created by duplicating meshes and displaying them through a render texture using a secondary camera.

The replay system then interpolates between positional and rotational values stored in a queue. On top of this, each object that is stored in the queue is capable of performing unique actions such as spawning particle effects inside the replay system.

## Abilities & Overloads

The player has a main weapon attached to their robot, which can inflict damage to other players. On top of this, crashing into other robots is also an easy way to inflict damage to others. Despite this, another degree of gameplay comes in the form of Abilities and Overloads.

### Abilities

Abilities are actions that can be taken to put the local player in an advantageous position or remove them from a dangerous position. These include abilities such as flipping the player upright or boosting the player forwards to remove them from danger.

Abilities have a mandatory cooldown before they can be used again, which is unique to each ability. In addition to this, some abilities can be used an infinite amount of times, whereas others may only be used a set amount of time before the player must acquire more.

### Overloads

Overloads are additional attacks that can be used on other players. These additional attacks include firing a homing missile at a player or detonating an impulse grenade on them to flip them over.

When using an Overload, a crosshair will be displayed on the screen which will automatically lock on to targets within a certain range. Once a target has been locked, the player is able to tap the attack button to initiate the overload. Once used, the overload will be consumed.

The player is able to iterate through their current abilities and overloads by tapping the Y and X button, respectively. Abilities and overloads are activated by holding the respective button.

Both abilities and overloads are spawned throughout the arena at an interval calculated in respect to the number of players in a match. Players must then race to collect them, as once collected they will disappear.

## Killfeed Interface

To increase immersion and keep players informed, a Killfeed Interface was created. This interface can be used locally or synced through the network on a case-by-case basis. The Killfeed will inform players when actions have happened in the game, such as when a player has been eliminated or when an ability/overload has spawned.

## Obstacles

Different obstacles have been placed throughout the arena in The Mayhem Pits. These obstacles provide a potential risk factor to the player, and also increase concentration needed to be successful in a match. These obstacles can inflict damage upon the player, or simply be a hinderance by veering them off-course or flipping them upside down.

## Database Functionality

The Mayhem Pits is fully integrated with Firebase’s Realtime Database Unity SDK. This allows players to create an account and log in. Creating an account on the database allows the user to access the full features of MyRobot, whilst also storing data such as available coins and tournament wins.

Database integration provides flexibility to the player, so they can log in to their account from any device and continue their previous progress in the game. As an added measure of ease-of-use for the user, creating a database account is not required to play the game. This means that the user is still able to connect to the multiplayer services and play the game with a basic robot and creating a database account only enhances the experience that they have.

## Multiplayer Services

The Mayhem Pits is a multiplayer-only game as it does not offer a singleplayer campaign mode. The drawback to this is that their needs to be multiple users playing the game at any given time for a user to find a match.

Multiplayer is provided through Photon Unity Networking, a service available through Unity’s Asset Store. This makes use of Unity’s core uNet services but extends its functionality and provides an easy to use library for the developer.

The player is able to join a match of two, three or four players. Once the match has started, no other players may join; even if another player has left. This stops any player from having an unfair advantage.

# Technical Structure

When the client loads, initial scenes will perform specific backend actions such as connecting to multiplayer services and checking for previously used database accounts. Once these actions have been performed, the user will be greeted with the main menu. In the main menu, the user has the option to change settings, enter MyRobot or join a match.

When joining a game, the player will be placed in a lobby with other potential players. Once all required players have joined, a countdown will begin, and the game scene will be loaded.

When the player’s game scene loads, a flag will be sent to the Master Client to inform it that this client is ready. Once all clients have registered as “ready”, a countdown begins on each client for the match to start.

If eliminated, the player is informed via the in-game interface. They then have the option to spectate the remaining players, or return to the main menu. When an elimination occurs, the Master Client performs a check to see if a player has won. If this check returns true, the match ends and each remaining user is informed of the victor’s username.

Once the game ends, each user’s rewards are tallied up based on their ranking in the tournament, and how much damage they were able to inflict to other robots.

# Gameplay

When the user joins a match and the countdown has reached 0, player logic is enabled on all clients and the match begins. The player is able to accelerate and rotate their robot using real-time physics.

Colliding with objects in the arena, or other robots, inflicts damage to the player and plays a collision sound effect with a volume relative to the impact velocity. The player must do their best to avoid collisions as, although the damage is minimal, it will quickly accumulate and can hinder their chances at winning.

On top of this, the player must also avoid obstacles that have been placed in the arena. These obstacles include a timed fire-shooting obstacle, and a collision-based launch pad.

Whilst dealing with these issues, the player must also focus on inflicting damage on other players, whilst ensuring they take no damage themselves. The player can use their main weapon to inflict constant damage to other players.

To increase strategic playing, weapon usage is based on a resource system as each weapon uses either electricity or fuel. These resources deplete over time, and slowly regenerate as the player is not attacking.

# Code Structure

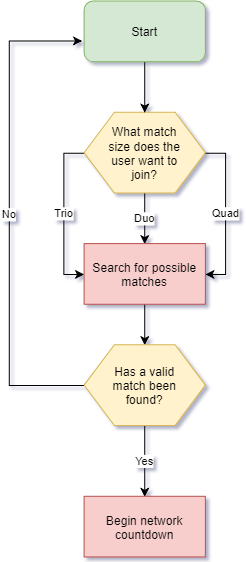
## Logic & Patterns

The Mayhem Pits is designed using an Object-Oriented structure. This structure makes use of inheritance and polymorphism to extend existing features easily. On top of this, as the game is designed in Unity Game Engine, the game makes use of a component-based design that Unity provides. Component-based designs allows for quick and easy modification of behaviours on-the-fly.

In addition to these structures, design patterns such as the Singleton Pattern is used frequently throughout the logic of the game. This pattern allows for global access to an otherwise private monobehaviour component. The singleton pattern also deals with potential conflict issues when multiple instances of the same class are in a scene at the same time.

These structures and patterns ease the development cycle of a project and provide a certain degree of future-proofing to a game’s logic. Below, logic diagrams have been provided which outline how the client deals with certain situations.

## Matchmaking Diagram



## Elimination Logic

