CloneFall 2: Technical Document

Introduction

This document is intended to illustrate the technical design, techniques, technologies and pre-existing templates used, as well as detail the mechanics, coding practices and flow of data in the game.

1. Overview

The team intended to recreate certain systems of the game: Titanfall 2. The team put focus on the movement mechanics, first-person shooting and basic networking to provide a multiplayer experience.

The development team, ‘Clone-Group-One-4’, consists of the following members:

- Erin Harper 2445245
- Bruce Tonkin *insert student number*
- Jean-Francois Retief *insert student number*

2. Prototype Objectives and Requirements

The team’s main goal for the prototype is to have a playable multiplayer prototype. Barring that, our minimum viable product is a test scene which can be used to illustrate the implemented movement mechanics and our gunplay.

So, to achieve our minimum viable product, the team needed our movement mechanics. These consist of wall-running, sliding, mantling, crouching and double jumping. The team will also need gunplay, which requires code for reloading, hit decals for feedback and aiming down sights, along with a basic hitscan registration of where the bullets will go.

To achieve the design goals, the team will need everything from the minimum viable product along with a UI, an arena for our players and -most importantly- some basic net code.

For the net code, the team is utilising a third-party framework known as Fishnet. This should drastically reduce the workload of getting multiplayer working, but utilising it will most likely still be a challenge due to the complexity of integrating it properly with our systems. Because of this, the team plans on integrating it as one of our first steps, so that future systems that can’t be run purely client-side can have the necessary integration built in for our specific purposes as the team continues development.

3. Technologies and Methodologies Used

3.1. Hardware Used

- Dell G15 5511 Laptops with an Intel® Core™ i7-11800H (11th Gen) processor and 16GB RAM (provided by Digital Arts Department of the University of the Witwatersrand)

3.2. Software Frameworks Used

- Game Engine:
  - Unity
  - Editor Version: 2020.3.30f1
- Source Control / Collaboration Tools:
  - Github
- Communication tools:
  - WhatsApp
  - Discord
- Documentation tools:
  - Google Docs
- Networking:
  - FishNet

3.3. Additional Templates and Resources Used

- Various tutorials online, such as those from Dave on Youtube (Dave / GameDevelopment, 2022) were invaluable for explanations of the underlying mathematics for mechanics such as wall-running.
- Bullet-hole sprites sourced from pngimg.com (pngimg.com, 2023)
- The sprites for the weapon are pulled from Titanfall 2 itself (Respawn, 2016).
- Fish-Net was used for the networking (Fish-Net, 2023)

4. Feature Documentation

This section aims to detail each mechanic in the game in isolation.

4.1. Feature: Multiplayer Networking

The team used the networking functionality from the third-party extension Fish-Net to allow multiple players to connect to one host game using an IP address. Once connected, players can see each other and even perform mock battles with the shooting mechanics. This mechanic has been implemented with the idea that it could be expanded more fully in future.

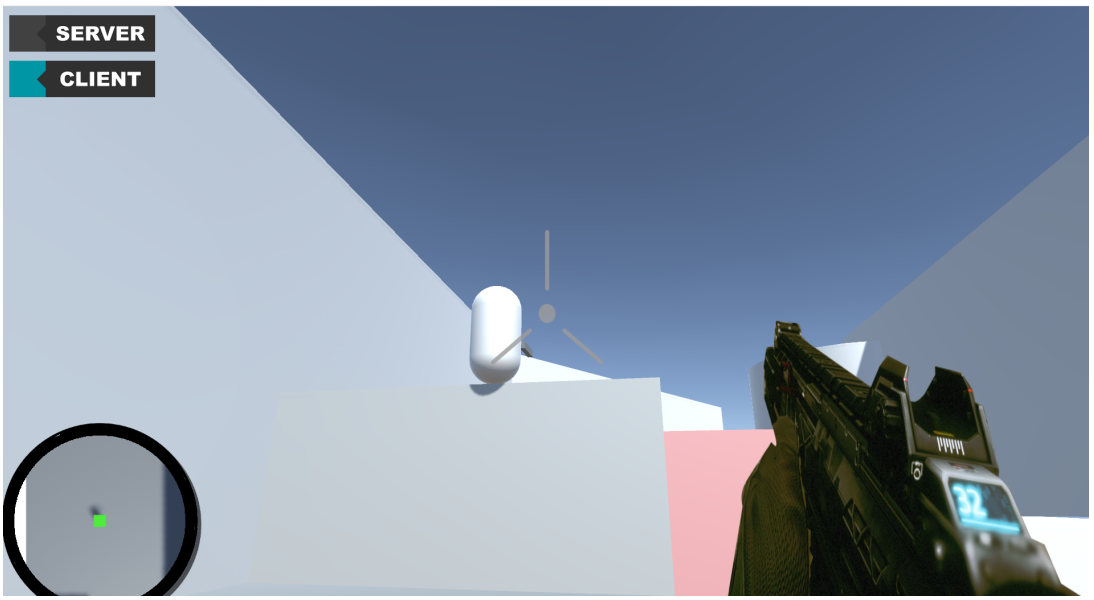

Figure 1: A look at two players in a scene together.

4.2. Feature: Movement

4.2.1. Base First-Person Movement

Simple implementation of the classic WASD movement, jumping, double jumping and crouching.
The movement is implemented with a rigid body, and the unity inbuilt physics system, not with a “character controller” component.

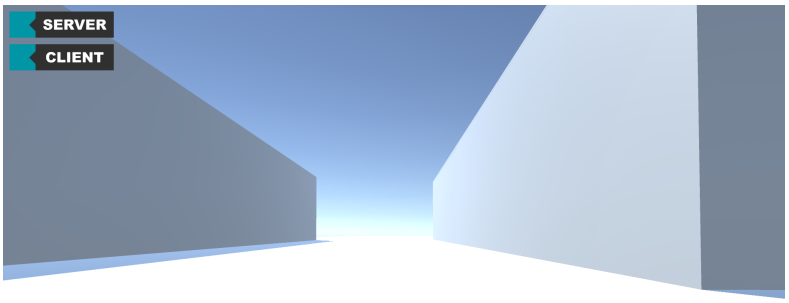

Figure 2: Image of the initial testing scene with two testing walls

4.2.2. Wallrunning

From the player avatar, two raycasts are sent out to the left and right of the player’s current orientation. While the player is in the air, these raycasts are used to check if the player is close enough to a wall to enter the “wallrunning” state.
While in this state the player will move tangent to the wall, with a reduced effect of gravity (relative to normal mid-air movement), until they exit the “wallrunning” state.
To exit the “wall-running state”, the player must either be grounded (reach the floor) or jump out of the wall run (press the jump key).

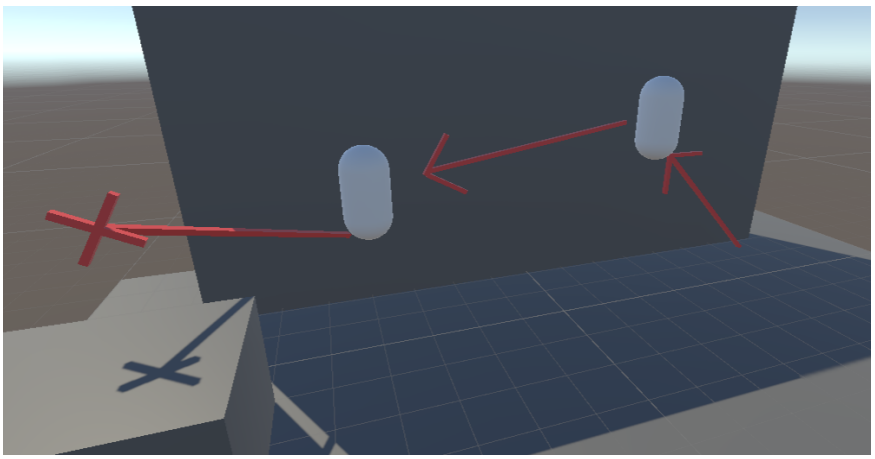

Figure 3: Illustration of wall-running and wall jumping

The main difference between this wall-running system and the system of Titanfall 2, is visual feedback, such as camera rotation. This was omitted due to potential conflicts with the multiplayer implementation (multiple camera instances, etc.).

4.2.3. Feature: Power-sliding

When the crouch button is pressed while the player is moving, a force is applied to the player's forward direction to make them slide fast.
Like crouching, it also decreases the player's length so that they can slide under certain objects.

4.2.4. Feature: Mantling

Two ray casts are sent from the front of the player avatar, one from their feet, another from their head. Only when the feet-ray cast hits an object, while the head-ray cast does NOT hit anything, an upward force is applied to the player avatar to allow them to mantle an object.

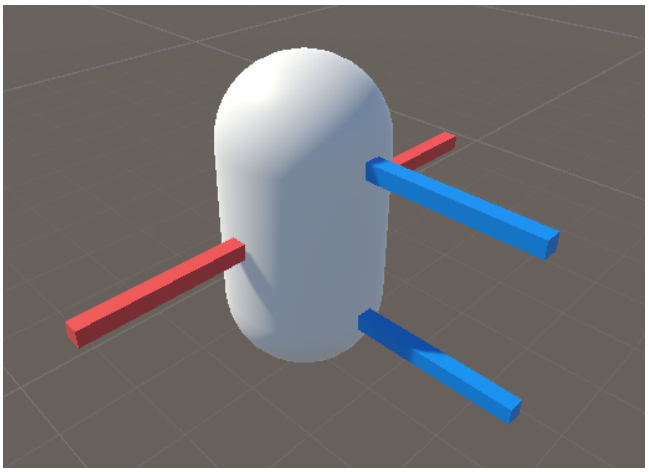

Figure 4: Illustration of base raycasts from player prefab (red → wall-running, blue → mantling)

4.2.5. Feature: Grappling

A raycast from the camera's position, in the direction of the camera's orientation, to a specified max length is used to check if the grapplehook makes a connection, if it does, a line-renderer connects the start and endpoints of the raycast and applies a jump force in the direction of the raycast.

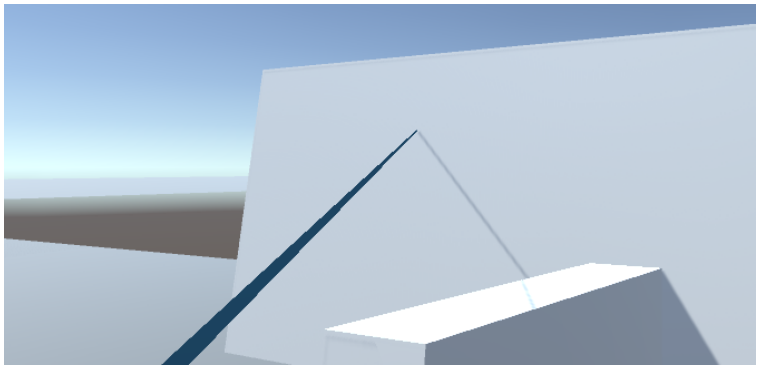

Figure 5: Image of a grapple line renderer, while grappling

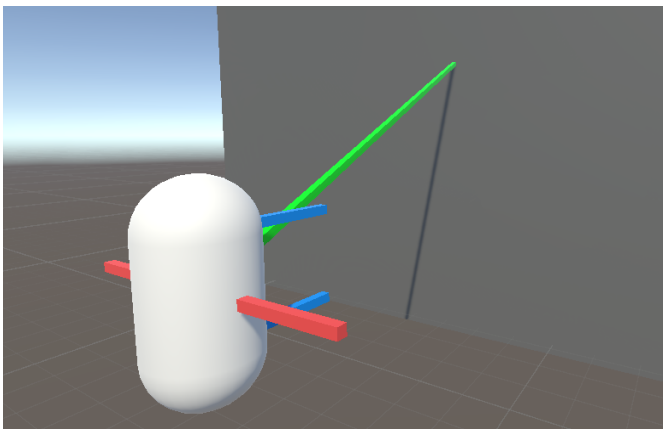

Figure 6: Illustration of grapple raycast (green → grapple)

4.3. Feature: Gunplay

4.3.1. Basics

The game features a ray cast-based shooting system that allows players to have precise and fast-paced gunfights.

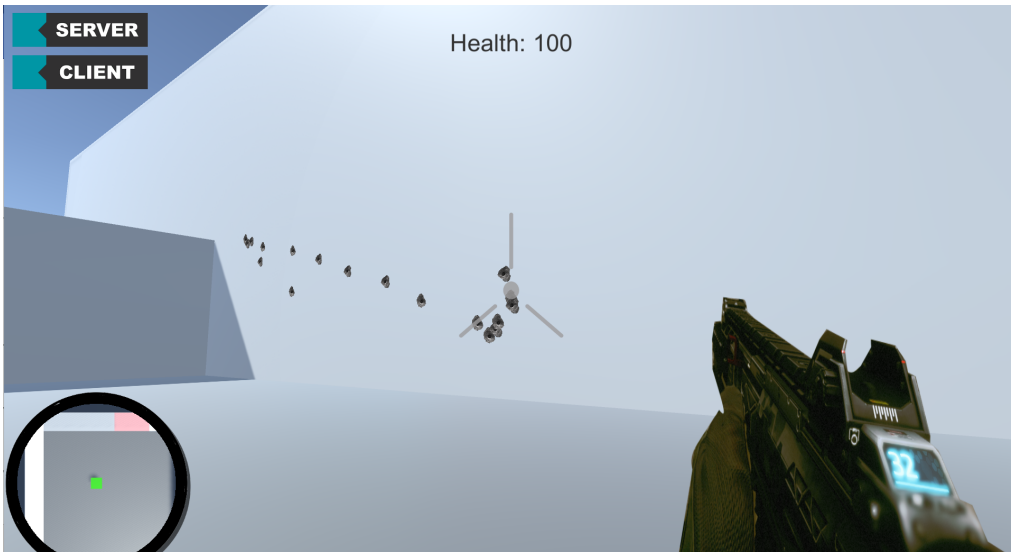

Figure 7: Bullet decals on a wall.

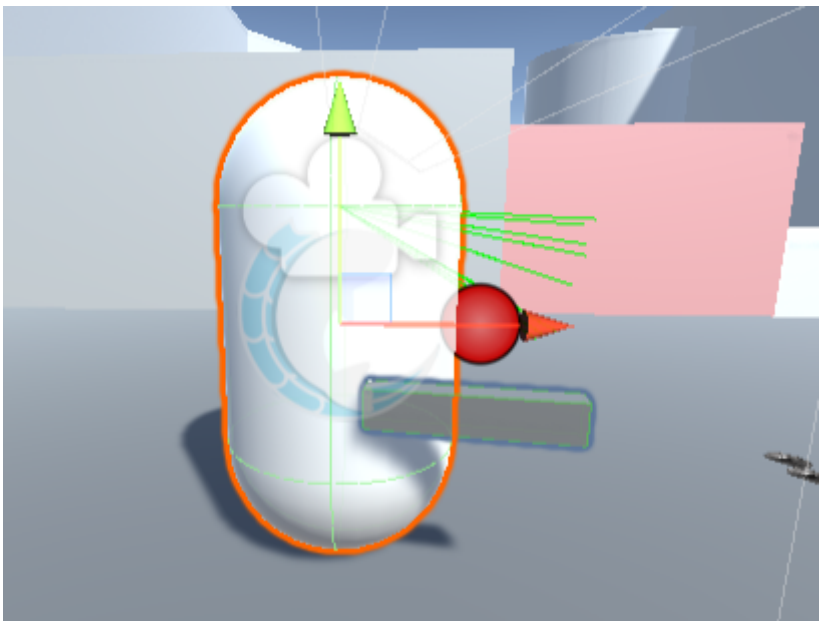

Figure 8: A visualisation of the raycasts

This system is bolstered by hitmakers and bullet decals, which improve readability for players and let them know where their bullets are going even beyond the reticle and sights.

4.3.2. Hip Fire random bullet variation vs. Aim Down Sight camera recoil

The game features two distinct modes of fire: aiming down sights and hipfire.

Figures 9 & 10: hipfire vs aiming down sights.

These two modes are mechanically distinct, with hipfire using a random calculation within a deviance range while ADS uses a simulated recoil-based system. These dramatically different underlying models were designed to imitate the way that the feel of Titanfall 2's gunplay changes between an older Source game style and a more modern FPS feel when aiming down sights.

4.3.3. Ammo and reloading

The game features a basic ammo & reloading system, with limited ammo per magazine, but with infinite magazines (since there are no ammo pick-ups)

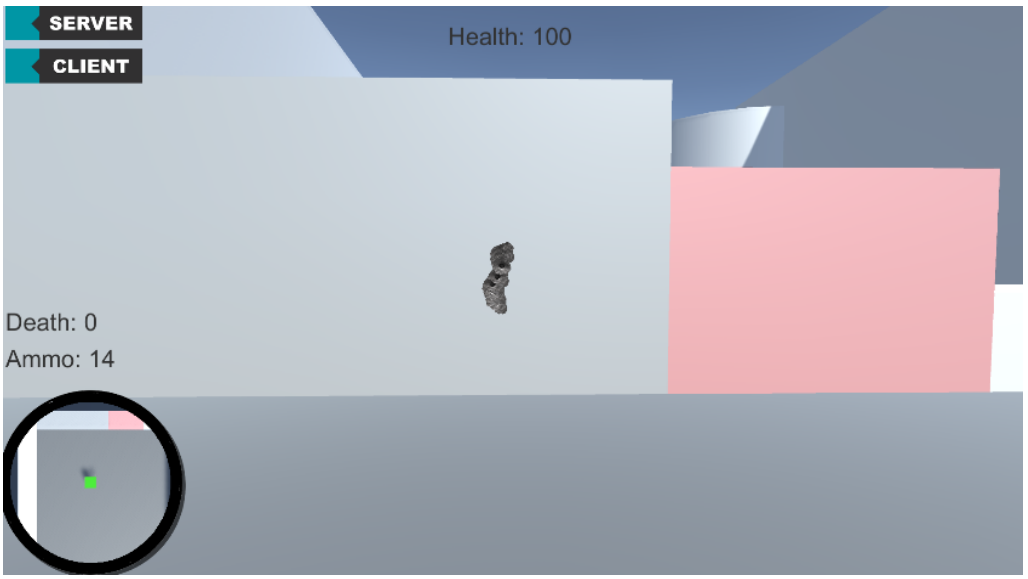

Figure 11: Ammo counter in the left, along with the gun down for reloading.

Our game features largely standard gunplay, especially because our gun is based on the R-201 (the starter weapon), but the team did add one mechanic not present in Titanfall 2: tactical reloading. In short, reloading with an empty magazine takes less time than with one that has bullets, which serves as a useful mechanic for if a player runs out of ammo in a gunfight. The team did this because it wasn't possible to implement the complex animation checkpoint system that Titanfall 2 uses for reloads with no animation, and because of that, our mechanic felt like it needed something to compensate. So, the team added a new mechanic from modern titles to lessen that gap.

The team also only uses a per-magazine ammunition system, the same as Titanfall 2. Titanfall 2 uses the lack of ammo reserve management to remove the weight of keeping track of reserve ammo from its already massive mental load, which The team also benefit from along with it reducing our development complexity.

4.4. Feature: UI

4.4.1 Menus

As the player starts the game this is what will show up so that they get excited and feel immersed in the gameplay

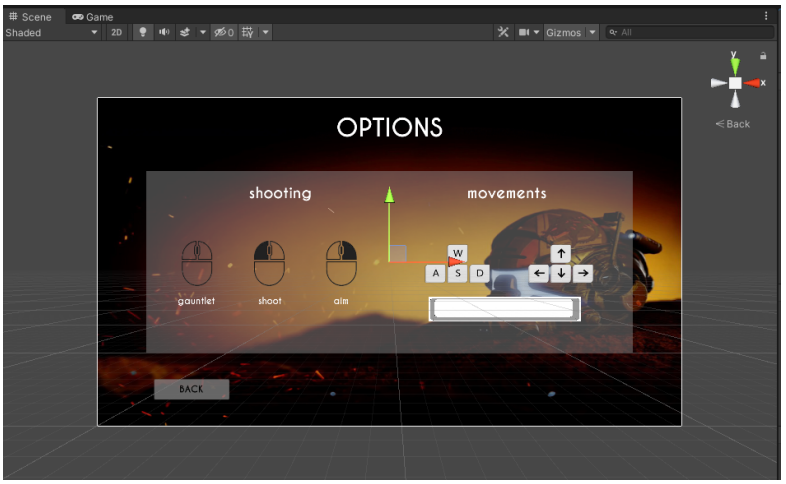

Figure 12: Options menu designed to show various controls to the player within the game, this is what you would normally find in most games

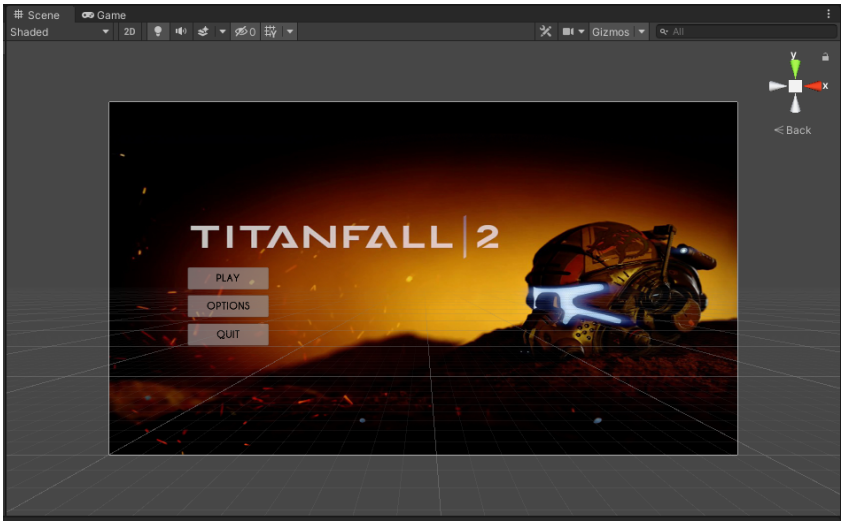

Figure 13: Main Menu screen designed exactly like Titanfall 2 homepage (image source: Titanfall 2)

4.4.2 Minimap

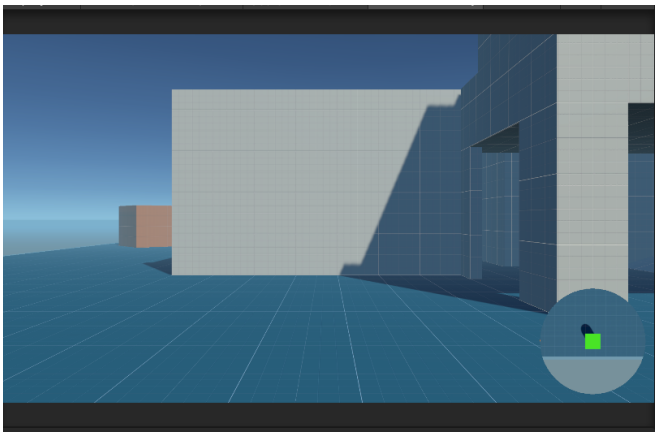

Figure 14: Demo version of Minimap before integrating it into the final game scene

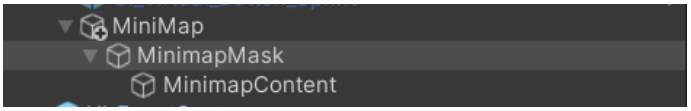

Figure 15: How the Minimap was created to be indicated to the player

A minimap in the left corner of the screens gives the player more information about the surrounding area, especially what is outside their field of view. It has also been placed there so that it does not interfere with the gun that has been put on the right-hand side of the screen.

Figure 16: Image of Minimap on display

4.4.3 Raspberry Death Screen

A raspberry death screen appears every time the player dies, as a form of indication to the player that they have died, a few seconds later there will be a pop-up written in words that tell the player what they will need to press in order to respawn. The spawn points vary, so no player will spawn at the same point twice.

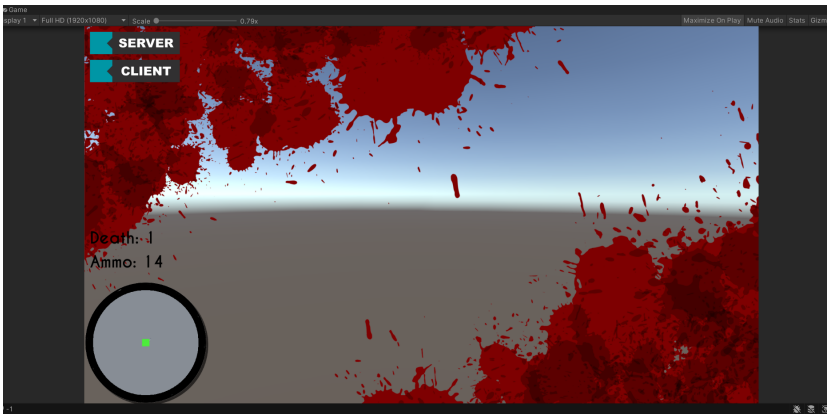

Figure 17: Raspberry Death screen that appears when the player dies

Figure 18: Respwan notification that appears a few seconds later

Figure 19: Screen showcasing the updated amount of times the player died (indicated by 'Death')

5. Bug log

Bugs are Still in the game at submission

- ❖ Infinite Grappling sometimes
- ❖ Powerslide stopping underneath an obstacle leads to the player's camera clipping through geometry
- ❖ The host does more damage than other clients, but the effect this has on balance is mitigated by another bug: the host can damage themselves occasionally while moving at speed and shooting.

6. Testing Documentation

Most playtests were mostly internal bug testing. For future projects a larger emphasis must be placed on external playtests to gather user feedback. The team struggled in particular due to a poor workflow since the team had to build every time they wanted to test. This did at least ensure they never had issues with the build running differently to the editor, but it still could have been optimised better.

7. User Guide / **README**

When entering as a single player:

- Just simply start the game
- Click on “Server”,
- Then click on “Client”
- Press [spacebar] when prompted to spawn in.

When setting up a multiplayer game:

- Connect all users to the same router
- The host:
    - Get your own IP address, using the command line `ipconfig`
    - Send IP addresses to other players
    - Start the game
    - Input IP address in IP selector textbox
    - Click off of the IP selector textbox until the IP address text is updated
    - Click “Server”
    - Click “Client”
- Other players:
    - Start the game
    - Input the Host’s IP address in the IP selector textbox
    - Click off of the IP selector textbox until the IP address text is updated
    - Click “Client” (AFTER the host has clicked “Server” and “Client”)

References

- Dave / GameDevelopment. (2022). Unity Tutorials. Youtube. Retrieved July, 2023, from https://www.youtube.com/@davegamedevelopment/featured
- pngimg.com. Bullet holes PNG images for free download, Retrieved July, 2023, from https://pngimg.com/images/weapons/bullet_hole
- Fishnet: Networking. Retrieved July, 2023, from https://fish-networking.gitbook.io/docs/.
- Respawn Entertainment. (2016). Titanfall 2. Available online: https://store.steampowered.com/app/1237970/Titanfall_2/