

# **Group Project Report: 3D Multiplayer FPS Game**

Maxwell Maia, Alexis Shaju, Albin Binu & Szymon Kozak

01/04/2023

--

CT216 – Software Engineering

--

Dr Enda Barrett

# **1. INTRODUCTION**

## **1.1. STATEMENT OF PROBLEM**

The objective of this group project was to create a 3D multiplayer first-person shooter (FPS) game that can be run in an internet browser. The project aimed to explore various technologies and integrate them into a cohesive and functioning web application. The game needed to be user-friendly and accessible, while also providing an enjoyable gaming experience.

## **1.2. STAKEHOLDERS**

The stakeholders for this project included the group members, Alexis, Szymon, Albin, and Maxwell, who all had different roles and responsibilities. The project's directed users were primarily gamers and the broader gaming community.

### **1.2.1. Goals**

- Develop a 3D multiplayer FPS game that runs in an internet browser
- Learn and integrate multiple technologies for game development
- Create a user-friendly and accessible gaming experience

### **1.2.2. Project Success & Evaluation Criteria**

- Successful integration of various technologies
- A functioning multiplayer FPS game that runs in an internet browser
- Positive grading feedback

## 2. USER REQUIREMENTS

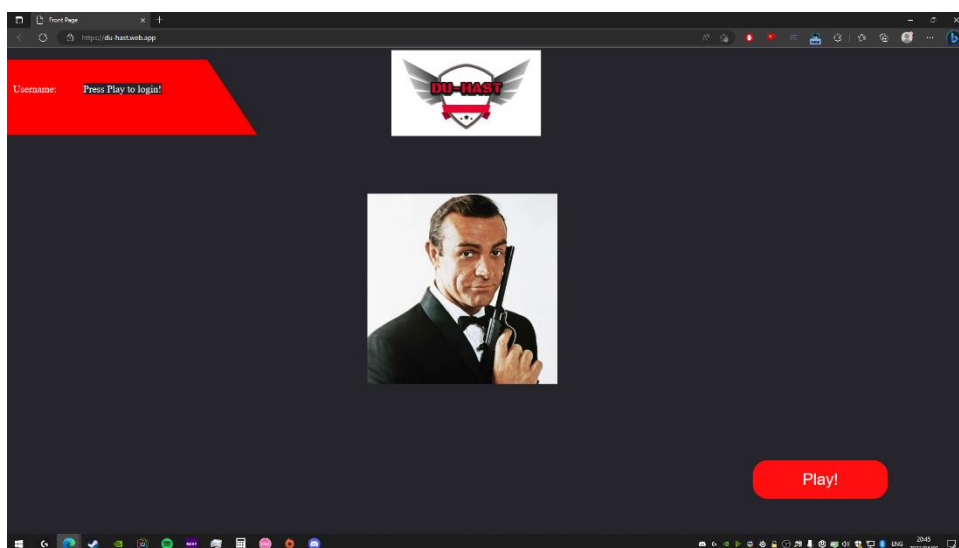
### 2.1. FUNCTIONAL REQUIREMENTS

- Players must be able to register and log-in into their accounts
- Players must be able to play a multiplayer game with others by creating or joining an online game room.
- Movement mechanics: WASD keys for movement, shift key for walking, space key for jumping
- Shooting mechanics: Left-mouse button for shooting, bullets, and magazine system. Reloading with r button replenishes magazine
- Player health and damage system
- Player death and respawn mechanics
- Audio and visual feedback for shooting mechanics

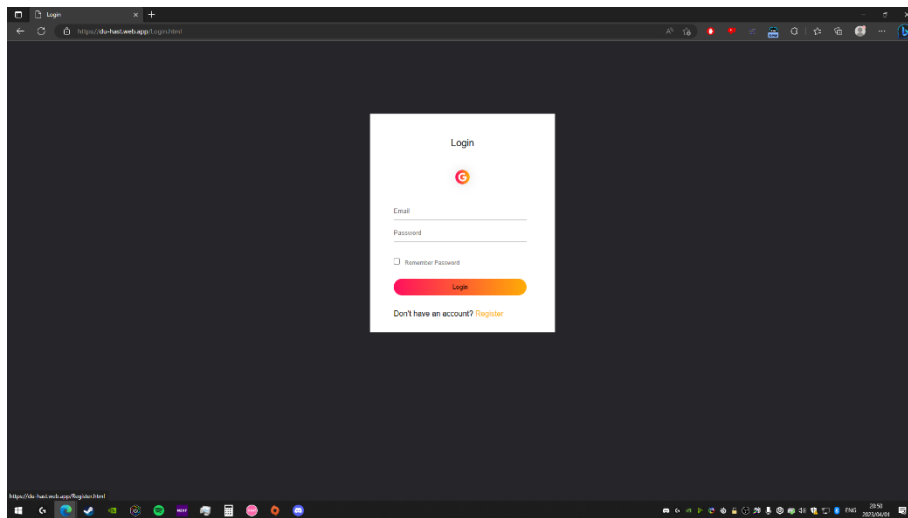
### 2.2. SCENARIOS

1. A user visits the game's website, logs in or registers, and joins a multiplayer game room. They can then play the game by moving around the map, jumping, shooting, and interacting with other players.

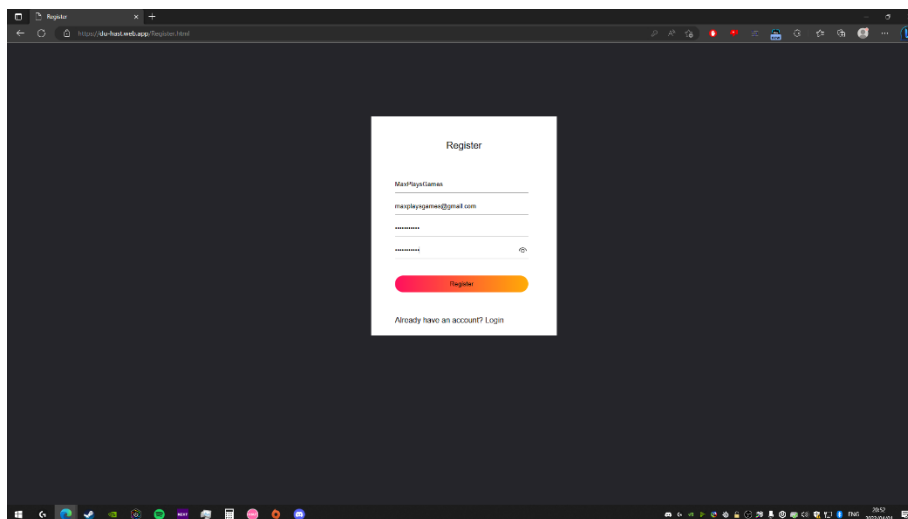
Visit the game's website



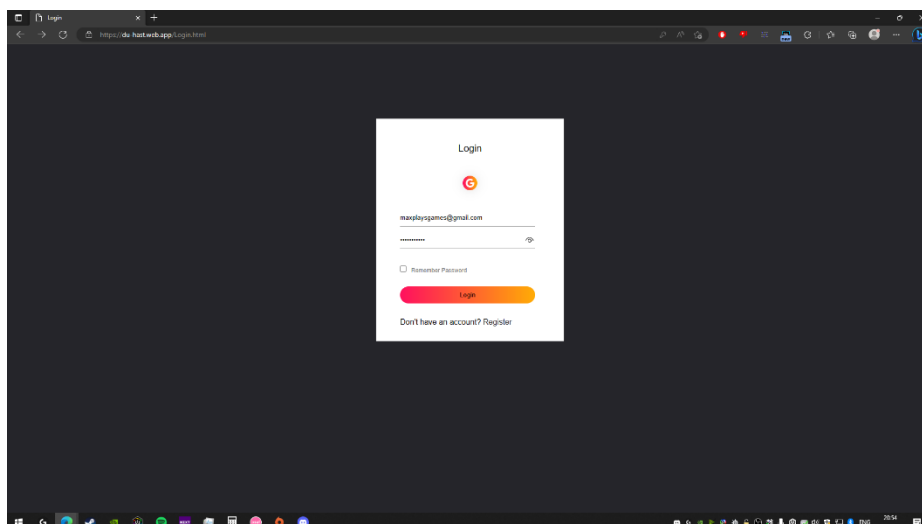
## Login or register



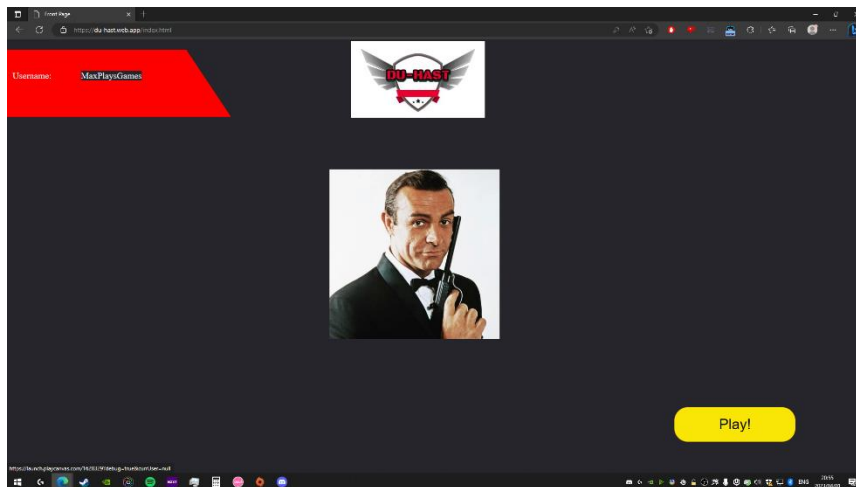
## Register



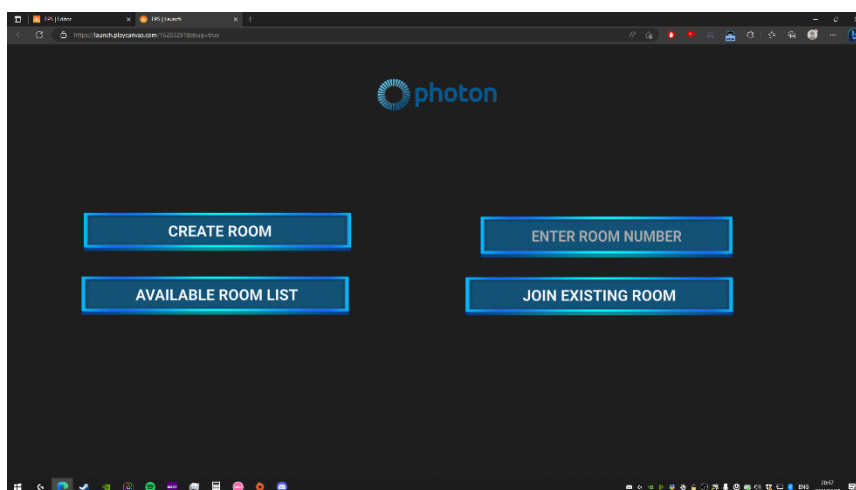
## Login



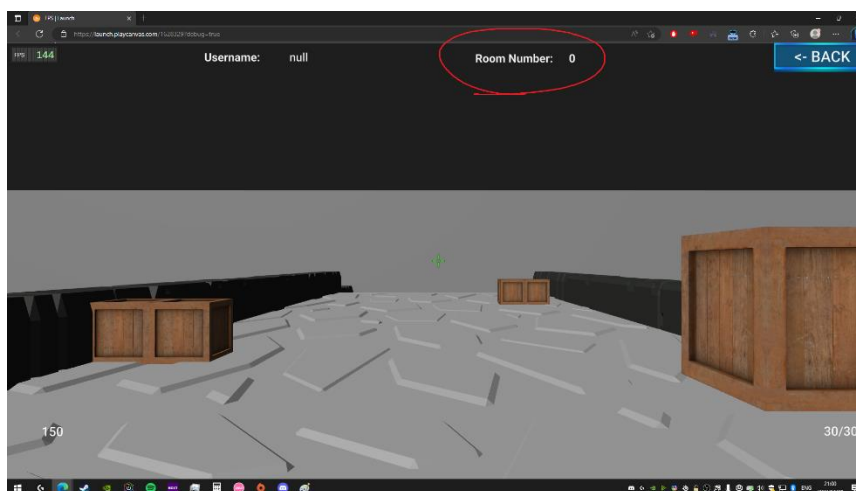
Username is visible on the home page after logging in  
Clicking play goes to the room selector



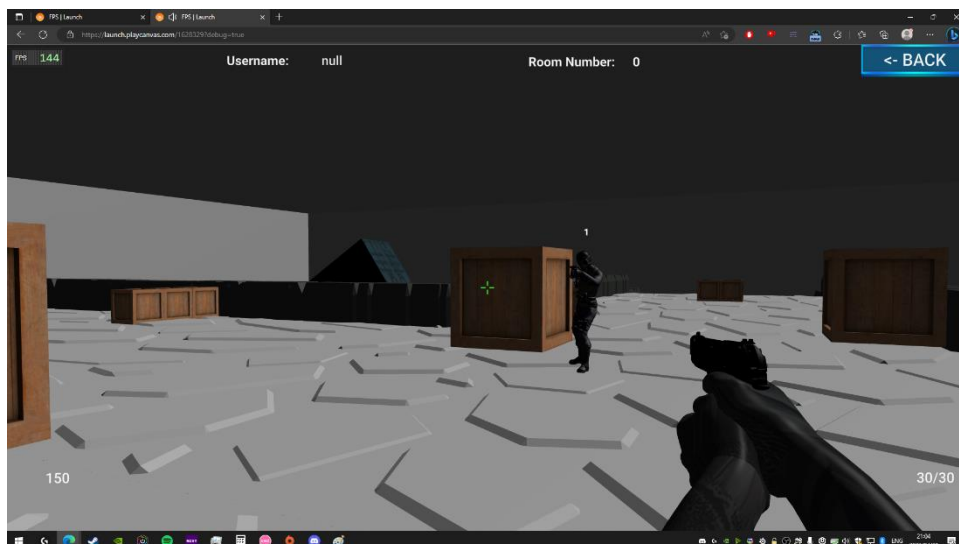
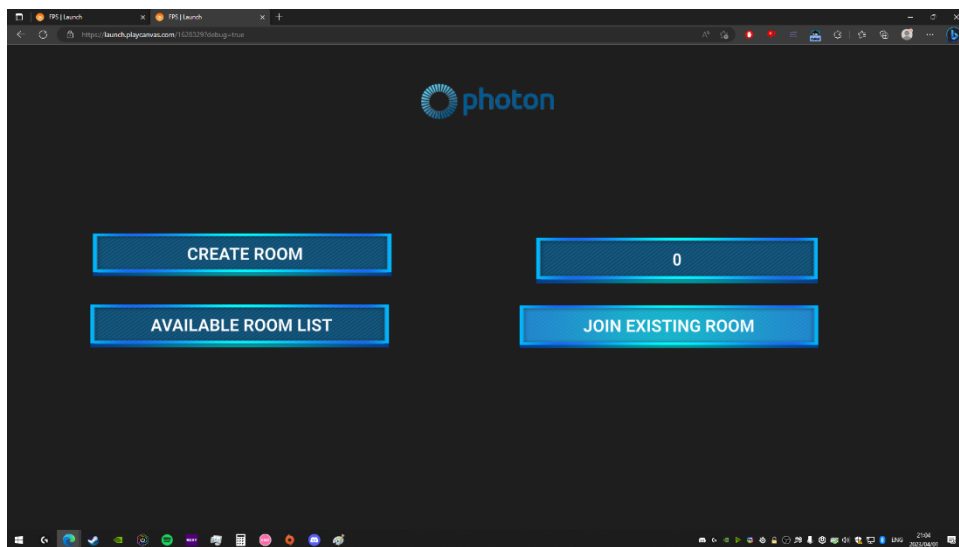
Room selector



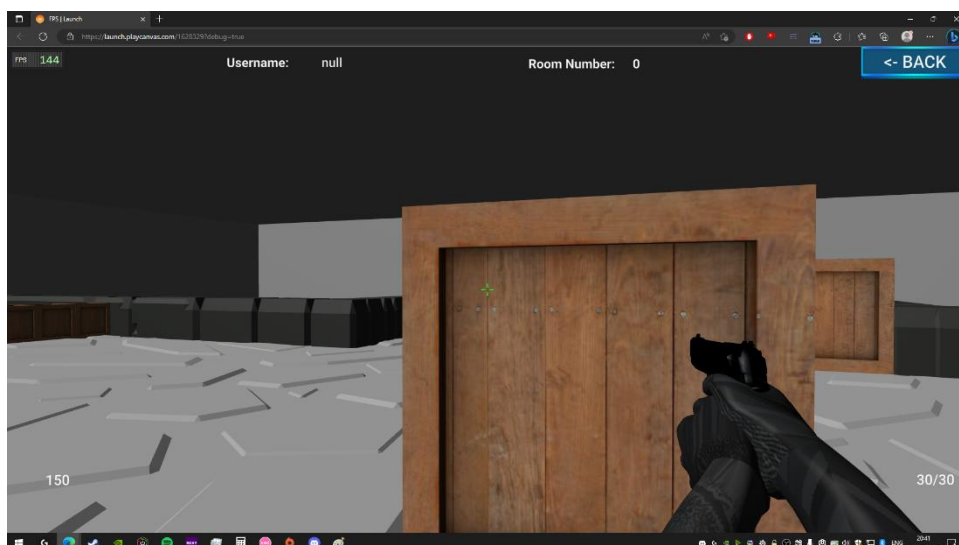
After the first person creates a room, a game room is opened



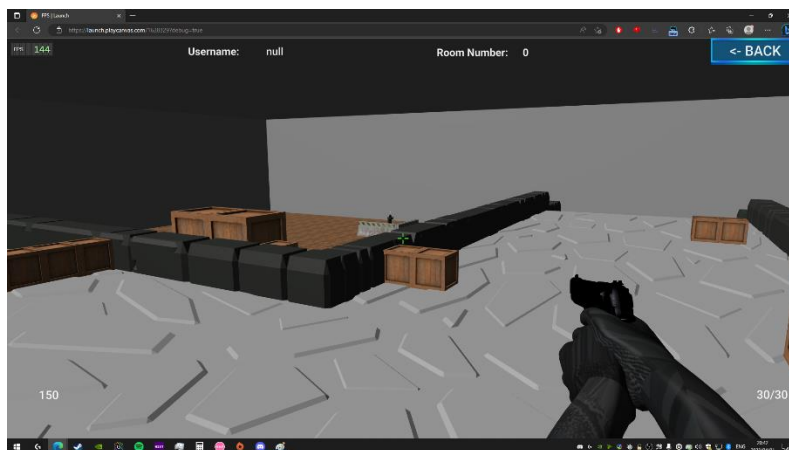
Friends can join by entering the room number



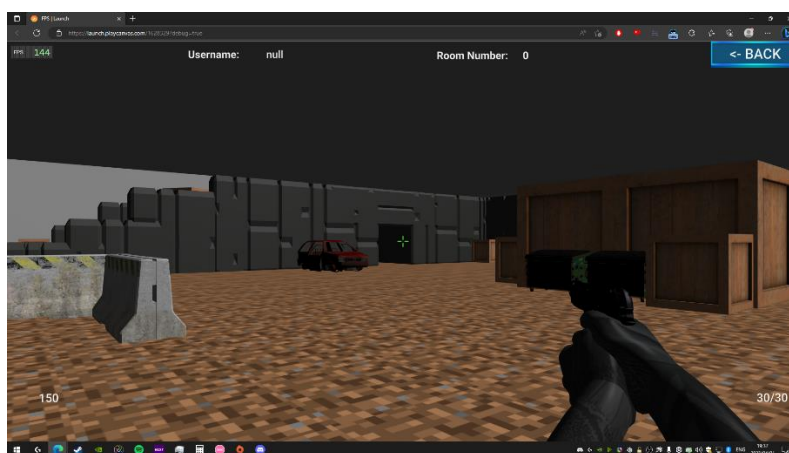
Can move around the map using the W, A, S and D keys



Can jump on boxes by pressing space



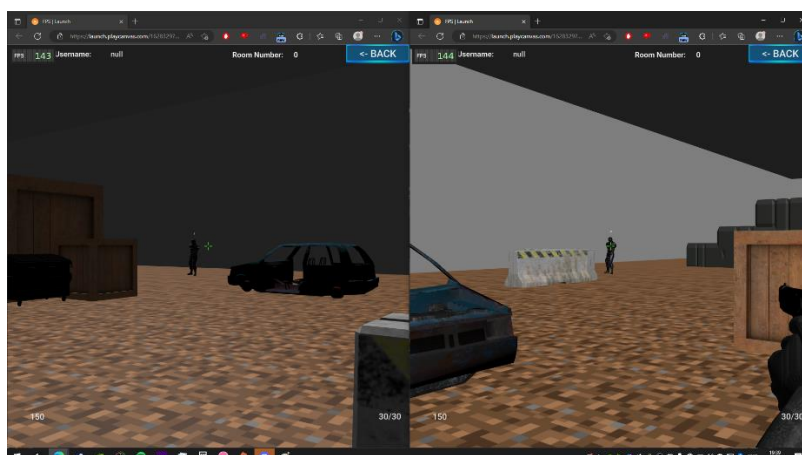
The player is free to move around the map



2. A user is playing the game and gets shot by another player. Their health is reduced, and they receive audio and visual feedback indicating that they have taken damage. If their health reaches zero, they die and respawn after a brief period.

Player 2

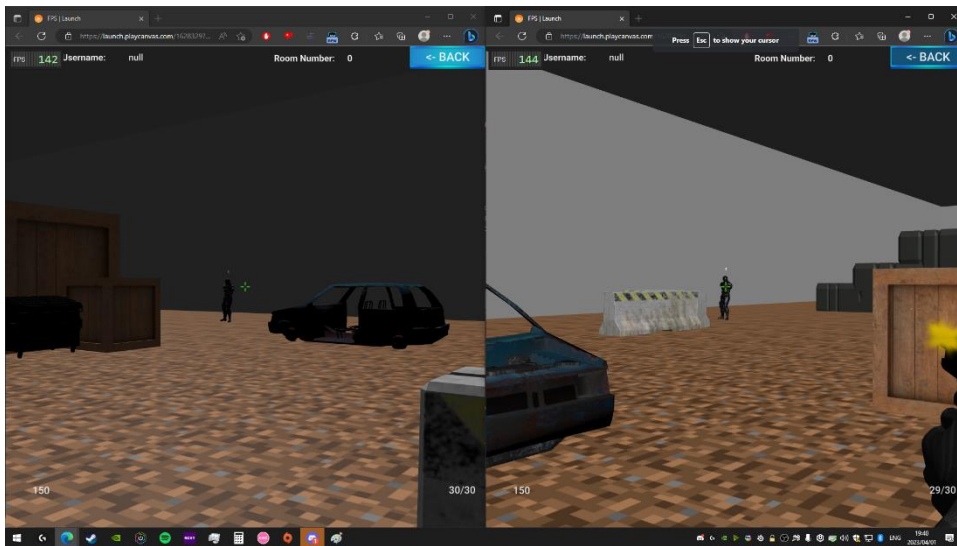
Player 1



Player 1 aims using their mouse and shoots Player 2 in the chest by left clicking:

Player 2

Player 1



Player 2 loses 40 hp. His health was 150 and now is 110.

Player 2



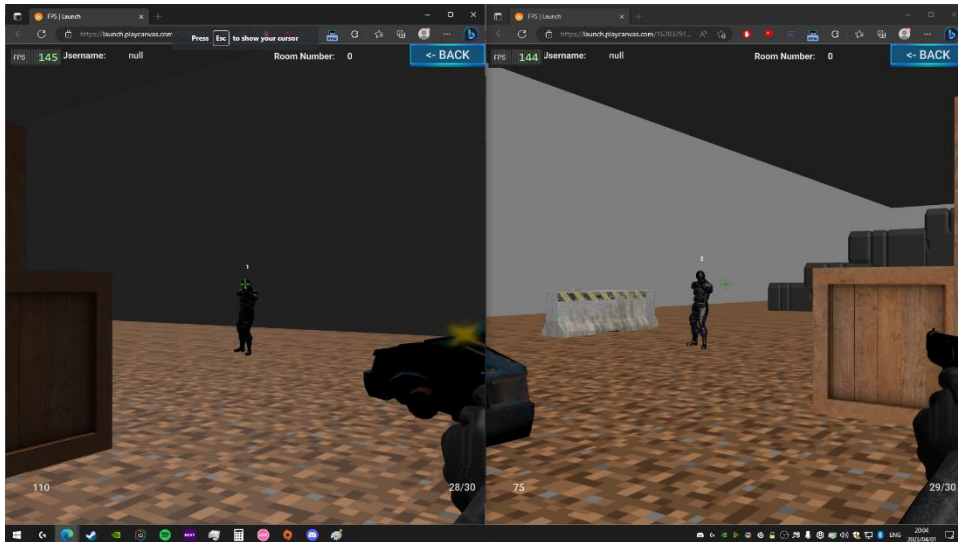


Player 2 moves closer to do more damage and aims at the Player 1's head to do even more damage.

Player 2 shoots player 1, dealing 75 damage.

Player 2

Player 1

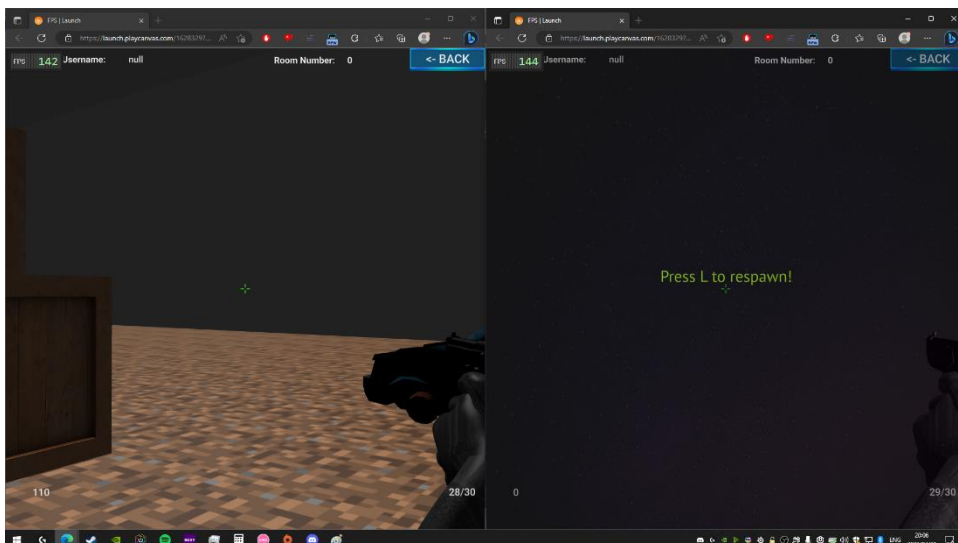


Player 2 shoots Player 1 again, dealing another 75 damage.

Player 1 is now dead

Player 2

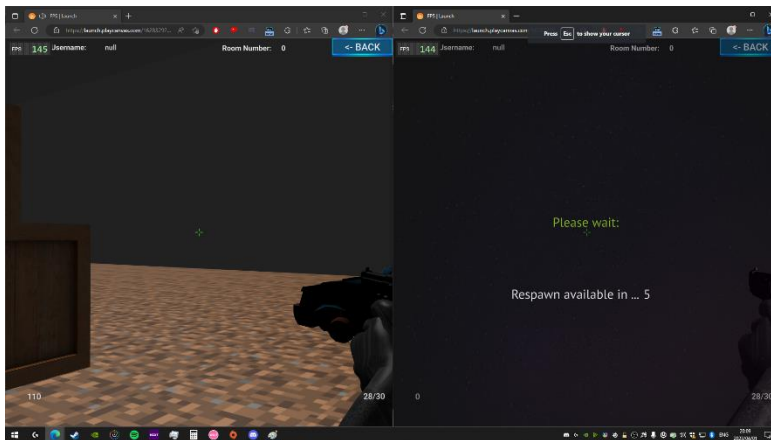
Player 1



Player 1 presses L to respawn and must wait 5 seconds

Player 2

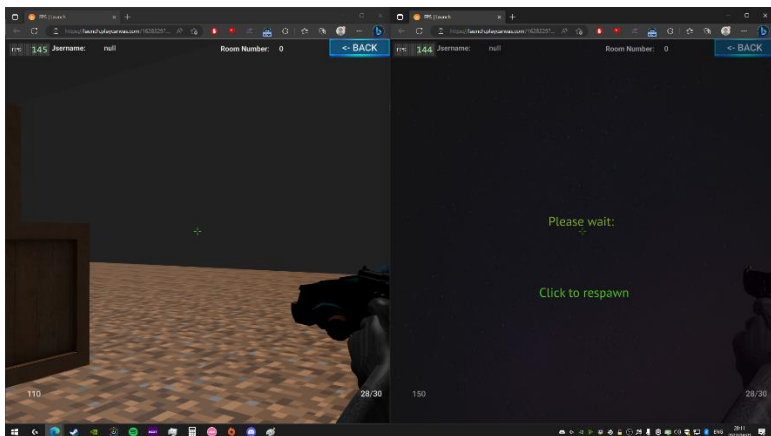
Player 1



After 5 seconds, Player 1 clicks to respawn

Player 2

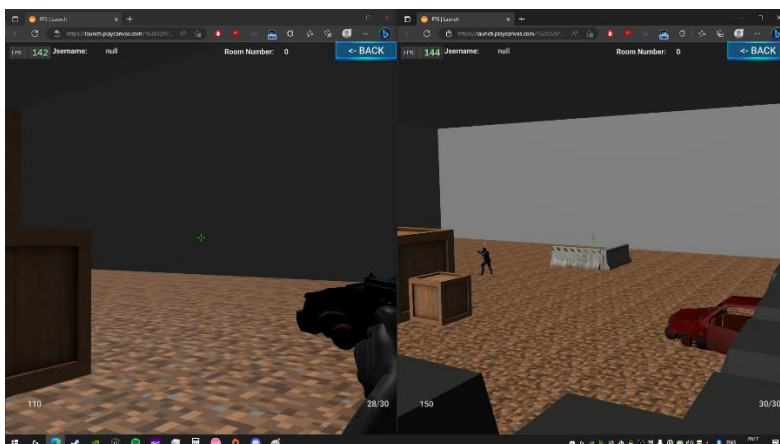
Player 1



Player 1 has respawned at a random location and is back in the game

Player 2

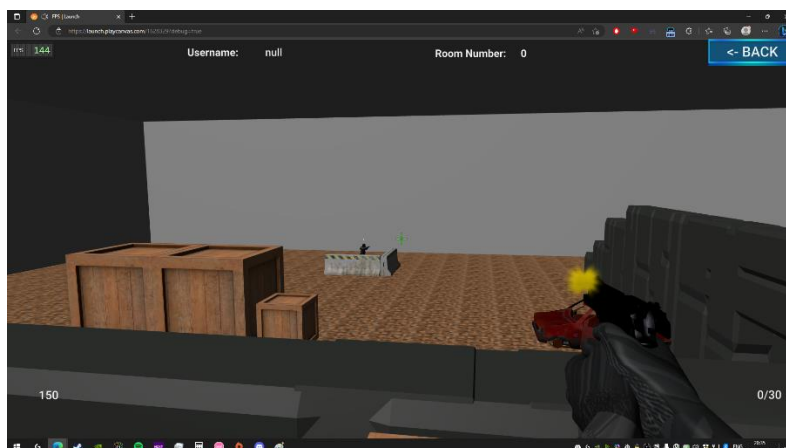
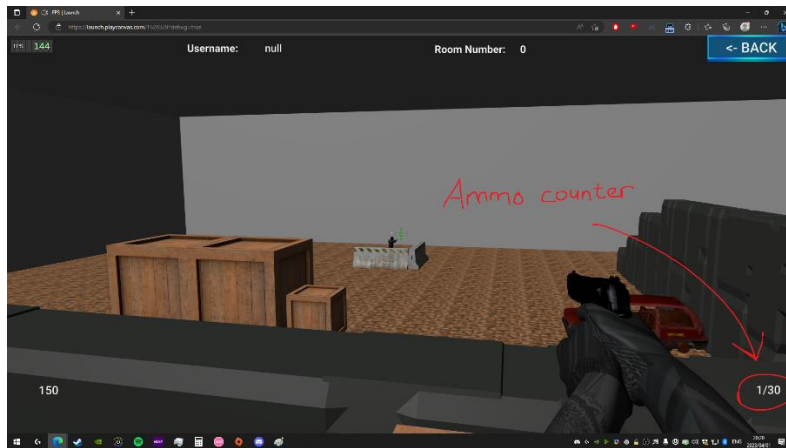
Player 1



3. A user runs out of bullets in their magazine while playing the game. They need to reload their weapon to continue shooting.

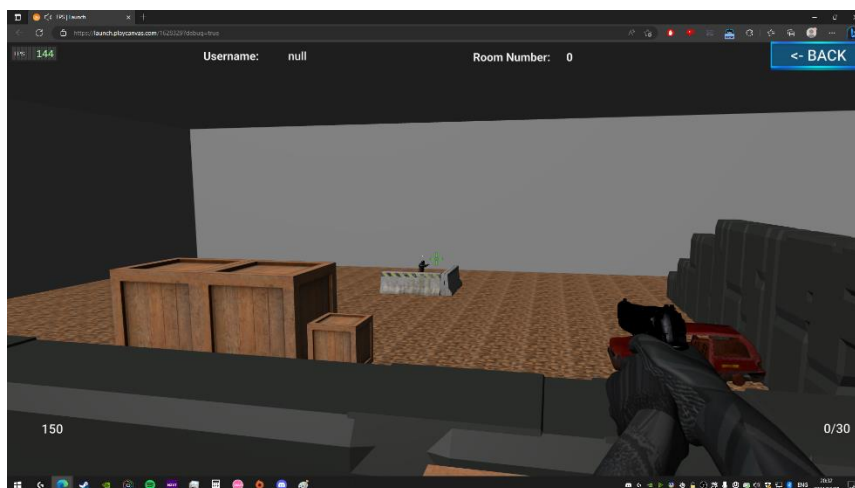
The player has 1 bullet left

Shooting now plays a gunshot sound, the gun recoils and a bullet is shot

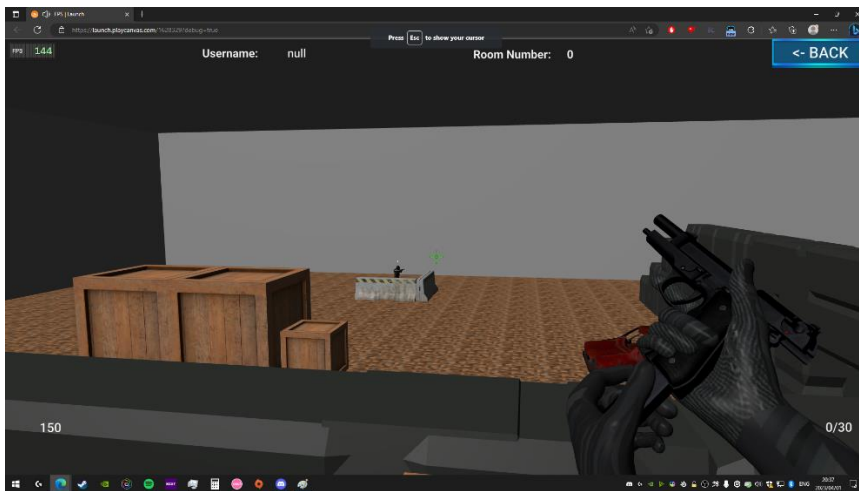


After shooting, they have 0 bullets left

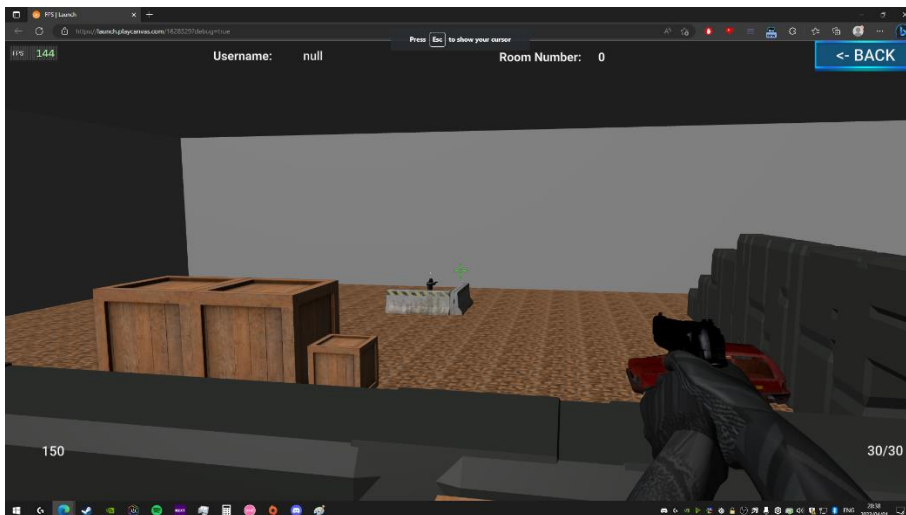
Shooting now plays the audible clack of empty gun and doesn't shoot



The player reloads by pressing R, their ammo is refilled after a reload animation



Ammo counter back at 30, the player can shoot again



## 2.3. PROTOTYPES

During the development process, several prototypes were created (in the form of PlayCanvas Scenes) to test and refine the game's mechanics, visuals, and overall user experience. These prototypes helped the team identify issues and iterate on the design, ultimately leading to the final product.

### 3. SYSTEM DESIGN

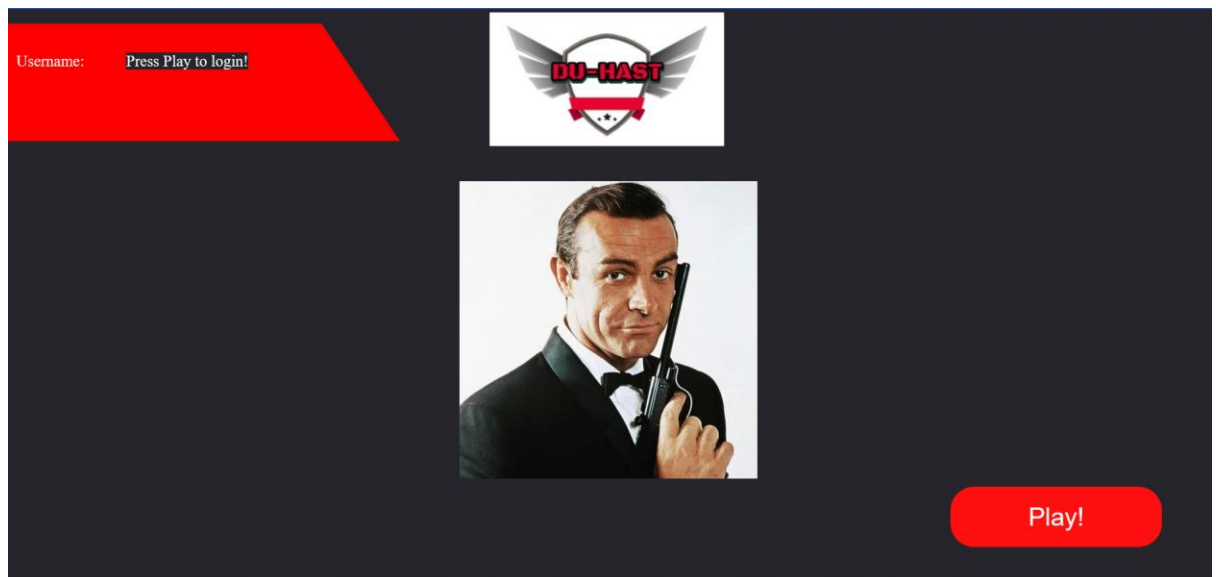
#### 3.1. ARCHITECTURE DESIGN

The game architecture consists of the following components:


- PlayCanvas: Game engine, code editor, and version control
- Photon: Multiplayer game hosting service
- HTML/CSS/JavaScript: Website front-end
- Firebase: Backend and authentication system

#### 3.2. UI DESIGN

- Website front-end: Including login, registration, and game access



Login



Email


Password


☐ Remember Password

Login

Don't have an account? Register


Username: Szymon Kozak





Play!

Szymon Kozak



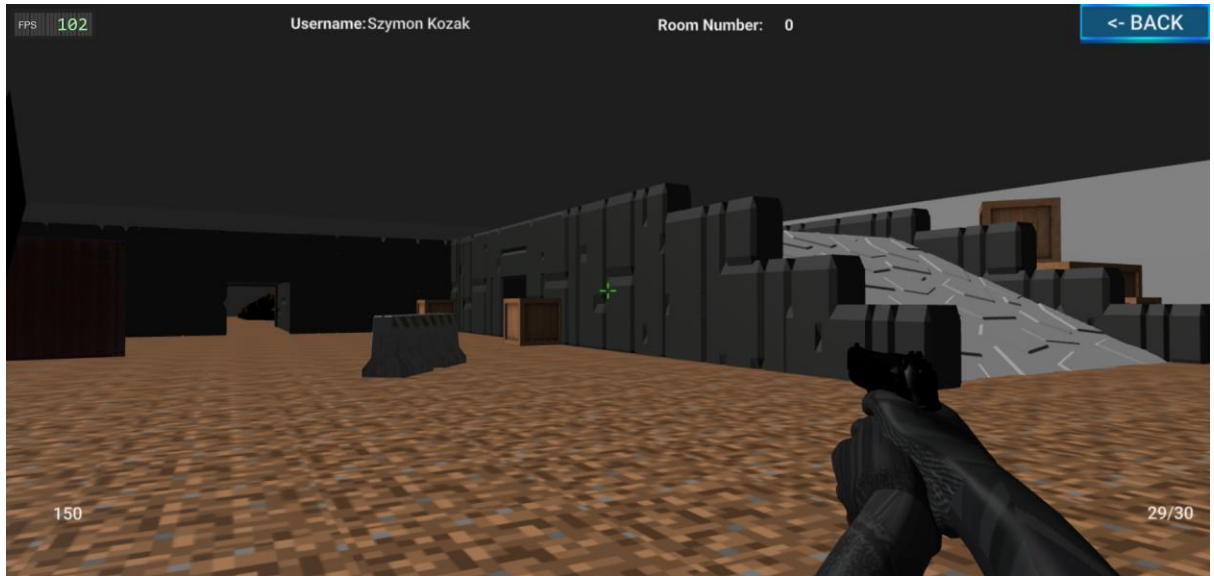
CREATE ROOM

ENTER ROOM NUMBER

AVAILABLE ROOM LIST

JOIN EXISTING ROOM

- In-game UI: Player Name; Health Point Counter, Magazine Capacity, Crosshair, Room Number; FPS Counter.



### 3.3. PROGRAM DESIGN

#### **Alexis Shaju**

##### Setup development environment:

The first thing I did, was search out a means to build this 3D FPS Game. We had first thought of learning C sharp or C++, with the idea of building the game in Unreal Engine or Unity. However, we were advised to build the game using JavaScript as this was the focus of the Software Engineering module. By doing some research, I found out about playcanvas, and how it was a very suitable engine for building projects ranging from simple websites to fully fledged games, even multiplayer. We decided to go with playcanvas. I set up the initial accounts and created a project. The interface of playcanvas was quite challenging in the beginning to learn, with all its quirks. However, we got used to it very quickly. While all this was happening, I looked into firebase authentication, and with the help of Maxwell, implemented authentication for our website, using email and even the option to login with google. Our main focus was to build the fps game, so we got the website part sorted out as fast as possible early on.

##### Implemented First-Person Movement:

With the help of Szymon, I worked to implement basic first person movement. This was one of the first sprint goals that we had, that we used as an opportunity to learn the basics about the playcanvas game engine, and scripting in playcanvas.



Although the code was written in Javascript, Playcanvas had a particular way of going about doing it. First defining the attributes, then variables at the first instance of the script, a part of the script that gets called upon every frame, and finally, any other functions we decided to add to the script. I myself, managed to implement the WASD movement, and Szymon tackled the jump and the shift walking element of movement.

### Photon Interface Creation:

Following the implementation of First Person Movement, I tackled the issue of which Authoritative Game Server we should use. We had the options of Colyseus and Photon, and we ended up choosing Photon. Colyseus code was written outside of the playcanvas script editor, on the local machine. Another problem was that it was written in typescript, meaning we would have to learn more about typescript, to tackle writing the code. Photon seemed like the better option. I first started by building a very simple project with Photon, trying to create two people in the same room. Here is the time, where I struggled very much with the project. My sample project with Photon, ended up being worse for me going forward, as I did not implement the logic to create rooms, view rooms and join rooms. This meant that every user was forwarded to the same room. Finally, with the help of the playcanvas team, and other helpers on the playcanvas forum, I was able to create a user interface to allow people to join rooms. The next problem soon came though. Other players, in the room, appeared to you as capsules. I was unable to represent them as player models, as I didn't know yet how to program this correctly. To fix this problem took at least 3 weeks, with me constantly researching how to instantiate player models, using photon and playcanvas. Finally with the help of the forums, I managed to overcome this hurdle.

Thanks to these problems I had encountered, I gained a fair amount of knowledge for programming using playcanvas scripting, which helped me going forward.

### Shooting Mechanics and UI:

With the help of Szymon again, I was able to implement shooting mechanics for our gun. We looked into raycasts, and getting information from raycasts, as to what object they hit etc. After Szymon mainly implemented this logic, I handled the UI elements of this, by importing in a gun model, and making bullet tracers to mark where the bullet landed after it was fired. The bullet was then destroyed moments after it hit the surface. Here Maxwell implemented first person animations to imitate a person holding a gun model, and different animations for fire, reload etc. I synced up the newly animated hand and gun model with the previous traces, so that the muzzle flash looked correct, alongside the tracers.



During this time, we figured it would be best, to receive a username for each player, upon clicking into the playcanvas website. To do this, we went back to our initial homepage that we built at the beginning. After our user signed into the webpage with firebase auth, when he hit play, we sent him to our game website. In this sent url, we included his game username, which was then picked up on the Playcanvas side and displayed to the user during the game.

### Game Mechanics and Features:

I then went onto tackle the hardest part of the project for me, in my opinion, which was the game logic itself. With the help of Albin, I worked out how to calculate the distance of a raycast, and based off the distance to change the damage. This meant that players closer to you, take more damage when shot, than players who are further away from you. Albin also created collision and rigidbody components for the player model, so that we could distinguish between a head, body and leg shot. I used these created components to implement this logic, assigning most damage to headshot, followed by body shot and finally leg shot. With the help of someone on the Playcanvas forums, I managed to correctly identify damaged players and update their UI based off that. Leading up to this, many problems were happening, for example, when I shot someone else, I was the one who took damage instead of the other person. There was also another problem, if there was 3 or more people in the room, if one person was shot, everyone except the person who fired the shot took the damage. To overcome this problem, another PlayerManager script was written. Our gunShoot script was run when a shot was fired. This script would send a message which was received by PlayerManager. If the players ID was equivalent to the ID of the entity that the bullet hit, then this player and this player only took damage.

After damage was working, player death and respawn caused problems. Maxwell had previously created two events for death and respawn which it was my plan to use. What I wanted, was that, when a player died, he disappeared off the screen, to everyone else, until he had respawned. However, whatever I tried to do, I could not get him to vanish off everybody else's screen. After much trial and error and debugging in the code, I found a sort of workaround. When the player's health was 0, I caused this entity to be disabled, so that it would disappear from the view of others. For the player themselves, on their screen, they were teleported to the middle of nowhere, so that they could not see the map, until they had respawned.

### Multiplayer Programming:

Most of the code which made this game multiplayer was in our player.js and game-room.js code. The logic behind it was, that each player emitted certain attributes about themselves, which includes, their position, their rotation, their animations, their health, if they were dead or alive etc. This was then taken in by game-room.js. What gameRoom did, was that it emitted these attributes of each player, and updated the room upon each frame. If a player had moved or rotated or even jumped, this was then smoothly shown to the other players in the screen.

## **Albin Binu**

### Home Page – Login/Register

The first priority I had was to create the home page. I took some inspiration from already existing web browser games like Krunker.io and Agar.io. For the login and registration, there were some difficulties when it came to positioning the button and the images on the page. I wanted to make the page more interactive, so I looked up tutorials to make the button more flashy. One thing from the login/register page that didn't make it to the final project was a sliding animation using CSS. This was a button between the login and register buttons. With the help of Alexis, To enhance user experience, our game offers an alternative registration method for new users. Instead of entering their email and password, they have the option to sign up using their Google account. This not only simplifies the sign-in process but also streamlines the registration process. By utilizing their Google account, users can conveniently import some of their personal information such as first name, last name, email, and profile picture. However, users have the flexibility to update this information if they desire.

### Map

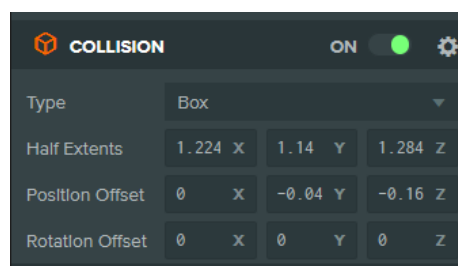
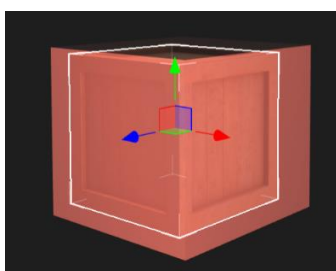
The map had many difficulties. First, I need to find all of the assets needed to make the map and their corresponding textures. The majority of the assets were reused from Maxwell's initial build. All of the assets used were found online using the Sketchfab website. I needed to make sure the assets were low-poly and free to use. Finding low-poly assets was essential; otherwise, it would have taken a toll on the fps and visual experience. The first model of the map had many problems. When I first tested it, the map was very laggy due to the number of entities in the game. There were around 200+ entities being deployed without including the players. This took a toll on the frame rates, which average around 20 fps, not very ideal. Now I had the problem of making the map more efficient. The main problem was the brick entities of the main building; this is where the majority of the entities came from. I ended up resizing and scaling each brick entity. And when

I re-ran the game, it had much better performance and stayed at a consistent 60 fps on my PC.



### Collisions for the Map

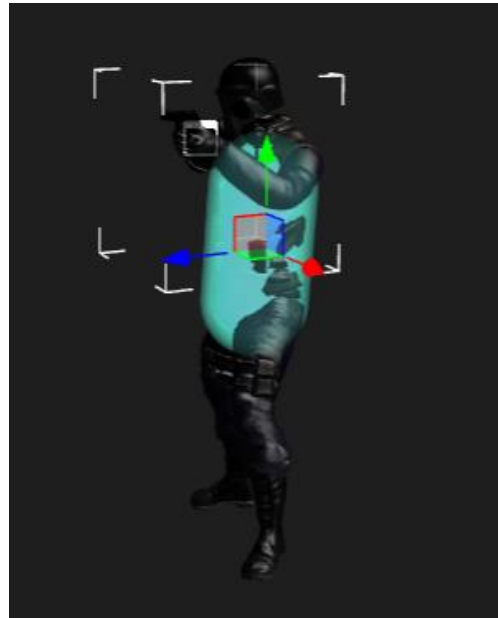
The next thing I had to take care of were the collisions and rigid bodies. The collisions were important, as otherwise the players could walk through the walls. The rigid body was implemented to each object to allow the bullet holes on the entities created by Szymon. This meant that each entity had to have its own specific collision and rigid body. It was annoying at first, as PlayCanvas offers very limited resources for the collisions, and there were only a handful of options to pick from. The main collision entity I used was the box. This meant that I couldn't use very obscure buildings because the collision entity could only fill so much. Only normal shapes like cubes and cuboids were used. Assets like trees and rocks had to be disregarded as they veered away from the normal shapes.



### Game Logic collaborated with Alexis

The next thing I had to handle was the game logic for the game with Alexis. This included the damage the enemy takes when shot at. We wanted to make the game authentic, so we decided to have the different parts of the body deal different amounts of damage. This meant I had to get different hit boxes for the different

parts—head, body, and legs. Like on the map, collisions only had limited collision entities, so the main entity I used was the capsule. It was annoying as the player model didn't have normal shapes (mainly curves), so some parts of the body didn't cover and others did. We tested this out the first time, and the collisions for the head weren't working. Each part of the body had its own tag. And when you shoot the person's head, it thinks you shot the body due to the overlap of the collisions.



### Spawn points collaborated with Maxwell

The spawn points were pretty easy to set up. The main code was done by Maxwell. My job was to put in the coordinates of the places where the player would spawn. The way I did it was I used a random entity on the map and moved it around to places that I thought were good. Then I entered the values into the array. Maxwell's code had an array of 12, so I implemented 12 positions on the map. I tested this using Maxwell's test code for spawning. When I press "L", it will automatically respawn me in a random location from the array.

### Scrapped player model creation

The last thing I tried but didn't end up making was our own player model. The idea at the start was to make a James Bond shooter with the James Bond character as the player model. I scoured the internet to find any models, but none were found. This meant I had to create my own. I tried to use Blender for this task. I had to put some time into learning Blender as I didn't have prior knowledge. I watched some tutorials on Blender to get used to it. After creating my first project, which was a donut, I noticed I wouldn't be able to use the software. Blender utilizes heavy

hardware, so my lower-end PC would crash when adding textures. So as a team, we decided to scrap that idea.

## **Maxwell Maia**

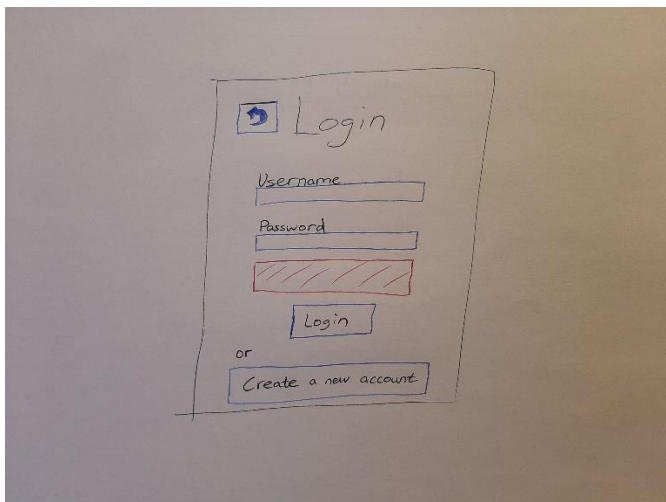
My duty was in the frontend development of the game. I was also the scrum master of our group. I managed sprint goals, co-ordinated scrums and created a group project Discord server for communication. I built some of the core components of the game using PlayCanvas's scripting environment. They were designed with the team in mind. My systems used event calls to initiate their functionality. This made it easy for my teammates to work with my code because after each component that I implemented, I posted a description of the event call, what it does and how to use it. I also created test scripts that helped me show the functionality of my components to my teammates during scrums.

### Designs

I created many initial designs for various aspects of our game. I created designs for the home, login and register screens. As well as a flow diagram. I created an initial large scale "map", the 3D environment that the player would be when they played the game. This inspired the final version.

### Initial designs

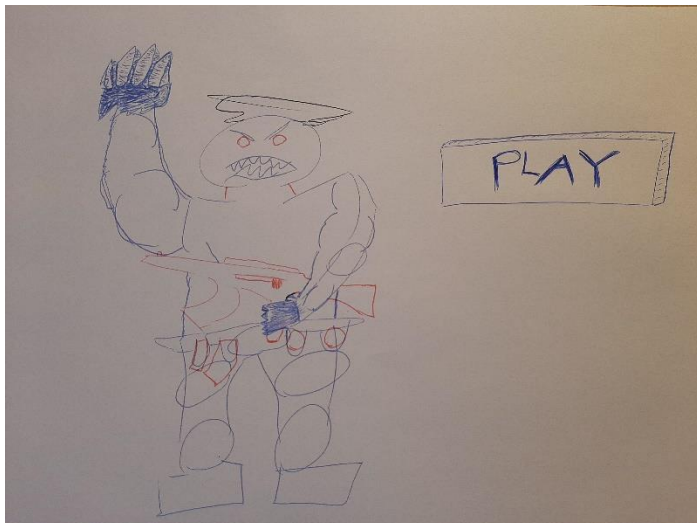
#### Login



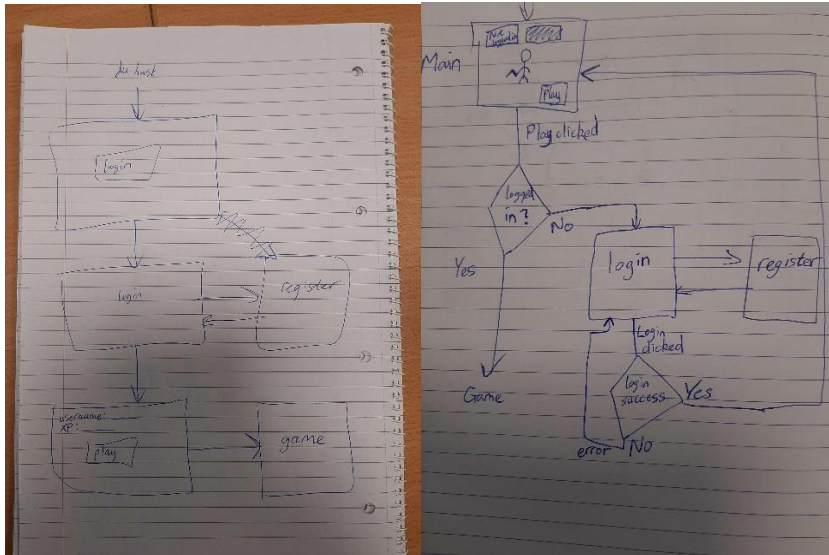
## Register

A hand-drawn sketch of a registration form titled "Register account" with a small icon of a person. The form contains three input fields: "Username", "Password", and "Re-type Password". To the right of the "Username" field is a "Check availability" link. To the right of each input field is a red hatched rectangular box, likely representing a validation or error message area. At the bottom center of the form is a "Done" button.

## Home screen



## Flow charts



Early map



During team meetings and through team discussions, I developed a word document with a realistic vision of what we wanted our game to look like. From there each member was able to pick features that they were most excited to develop.

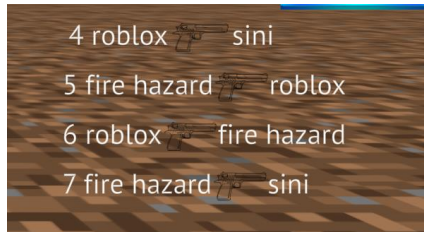
### Features that I implemented

#### **Kill feed**

I created a system that managed a kill feed UI. It allowed an event call to be made from any script in the program. The event call acted as a function. All it needed was the two player names as Strings and it would create a kill feed entry in the killfeed at the top right-hand side of the screen. See figure below. My script managed the UI components needed to display the text along with a picture of a gun indicating a kill. The script made sure that the most recent kill entries were added to the bottom of the kill feed. Each kill entry was on the screen for 5 seconds and the kill feed would dynamically update as new kill entries were entered



or expired. The kill feed appeared on every player's screen showing them who killed who in the game. This did not make it in the final version of the game.



## Death UI

I created the script and UI that the player sees when they die in game. Their vision becomes blurred and they see text that says "Click to respawn" along with a 5 second timer. Once the timer reaches 0 and the player has clicked to respawn the player is immediately back in the action of the game.

At the end of the respawn timer, my script called the respawn event which respawned the player.

## Respawning

I created an array of spawn locations that consisted of co-ordinated in the 3D map. I created a script that would set the player entity's position to one of the spawn locations, the script chose the spawn location from the array randomly. The script could be called anywhere in the program using an event call. I explained my system to Albin and he developed it further.

## Gun animations

I implemented the animations for the hands and gun. We found a free animation file online that perfectly suited our needs. The animations were not immediately compatible with our project. I faced a problem where PlayCanvas would play all the gun's animations instead of a specific animation. I created a script that would play the part of the animation that you wanted. I made an event call for each animation that other scripts used to play the desired animation. Animation event calls: shooting, reloading, reloading when the gun runs out of ammo or cancelling the current animation.



## Closing remarks

Working on this project was not easy. I had to learn how to develop 3D games in PlayCanvas from scratch as well as learn PlayCanvas's scripting. Including using PlayCanvas's built in source control. I learnt a lot about working in a team and gained valuable experience in Agile software development.

I also made myself available to assist others. Most notably I helped Alexis send display name data from the login page to the main menu page using JavaScript.

## **Personal Work Report: Szymon Kozak**

### Movement Mechanics:

I began my work by focusing on the movement mechanics of our 3D FPS game. This included implementing basic navigation using the WASD keys for forward, backward, left, and right movements. I also developed a 'walk' mechanic that slows down the player's movement speed when the shift key is held while moving, simulating a more realistic walking pace.

### Jump Mechanic:

Next, I created a jump mechanic that applies an upward force to the player model when the space key is pressed, simulating a jump. However, if the space key would be continuously pressed the player would end up flying in the air or 'leaping' indefinitely. Thus jumping could only be allowed while a user is planted on the ground and therefore I had to create a special tag for each ground surface that the player could stand on. Once I player model would collide with this surface, they would be 'grounded' and would be allowed to jump. This added another layer of interactivity and realism to the game, allowing players to navigate the environment more effectively.

### Shooting Mechanics:

My next task was to work on the shooting mechanics of the game. I began by implementing the primary shooting and reloading mechanics. When a player wants to shoot, they click the left-mouse button, which creates a raycast originating from the player model and extending towards the player's crosshair location in the game. If the raycast intersects with a solid surface, a bullet hole appears at the point of impact.

### Magazine and Reloading Mechanics:

I also created the magazine and reloading mechanics, including a magazine with a predetermined number of bullets that decreases with each shot. The game keeps track of the magazine capacity, and if the gun is empty, it will not fire. If the user presses the 'r' key the magazine is replenished

after a set timeout while adequate animations (thanks to Maxwell) and sounds are played in parallel. This feature adds an element of strategy and realism to the game, as players must manage their ammunition and decide when to reload.

#### Sound Effects and Animations:

Lastly, I incorporated sound effects and animations that correspond with the various shooting mechanics. These include the sound of a gunshot, the movement of the pistol slide, the reload animation, and an empty gun sound. By integrating these auditory and visual elements, I helped to create a more immersive and engaging gaming experience for the players. I carefully through trial and error, matched sound effect triggers times with the animation so that the user would be provided with seamless and truly immersive shooting experience. I ensured that the sounds would layer on top of each other in order as to prevent sound effects interrupting each other and break user immersion.

Overall, my work on movement mechanics, jump mechanics, shooting mechanics, and sound effects contributed to the development of a realistic and engaging 3D FPS game. By collaborating closely with my teammates and overcoming challenges throughout the project, we were able to create an enjoyable and interactive gaming experience for players.

## 4. CORE TECHNOLOGIES

### 4.1. IMPLEMENTATION DESCRIPTION

The game was developed using the following technologies:

- PlayCanvas: Game engine, code editor, and version control
- Photon: Multiplayer game hosting service
- HTML/CSS/JavaScript: Website front-end
- Firebase: Backend and authentication system

### 4.2. TESTING

Each group member was responsible for testing their respective components and features. The team also conducted continuous integration testing to ensure the smooth functioning of the game as a whole.

## **5. INTEGRATION & DEPLOYMENT**

### **5.1. INTEGRATION STRATEGY**

The team used PlayCanvas as the primary platform for game development, which allowed for seamless integration of game mechanics, multiplayer functionality, and UI design. Firebase was used for backend services and authentication, which was integrated with the website front-end. Photon, which uses authoritative servers, was used for hosting the multiplayer rooms. This server held all the game logic. It was sent information by clients, and updated the visual experience of all clients, with the information it received.

### **5.2. DEPLOYMENT PLAN**

The game was deployed on a web server and made accessible to users through a web browser. The team ensured that the game was optimized for performance and compatibility with different browsers. Prior to deployment, the game underwent extensive testing to identify and fix any remaining bugs or issues.

## **6. RISK ASSESSMENT & MITIGATION**

### **6.1. IDENTIFIED RISKS**

During the development of our multiplayer FPS game, several risks were identified. These risks could have potentially hindered the project's progress or negatively impacted the final product. The main risks identified were:

1. Inadequate knowledge of the technologies used: The use of PlayCanvas, Photon, and other technologies was new to the team members, which could lead to delays and difficulties in implementation.
2. Poor version control: The team initially experienced issues with version control, resulting in lost progress and conflicts when merging code.
3. Performance issues: The complexity of the game and the use of high-quality models could lead to performance issues, affecting the gameplay experience for users.
4. Scope creep: With many features and ideas, there was a risk of the project scope expanding beyond what the team could achieve within the given timeframe.
5. Security risks: The game required user authentication and personal information storage, which could expose vulnerabilities if not implemented securely.

### **6.2. MITIGATION STRATEGIES**

To address the identified risks, the team implemented various mitigation strategies to ensure the project's success and minimize the impact of potential issues.

1. Learning and research: The team members dedicated time to learn about the technologies used in the project, including PlayCanvas, Photon, and Firebase. This involved completing tutorials, consulting documentation, and seeking help from online forums when needed.
2. Improved version control: The team established a version control system with separate branches for each member's work and frequent checkpoints. This system helped prevent conflicts and allowed for efficient collaboration between team members.
3. Performance optimization: The team focused on using low-poly models, optimizing textures, and ensuring efficient code to minimize

performance issues in the final product. Regular testing was conducted to monitor performance and address any issues that arose.

4. Prioritization and time management: The team set clear priorities and goals for the project, ensuring that the scope remained manageable. Regular meetings were held to discuss progress, identify potential issues, and reallocate resources as necessary.
5. Security measures: The team utilized Firebase for secure user authentication and data storage, following best practices to protect user information.

## 7. CONCLUSIONS

Throughout the development of our multiplayer FPS game, the team faced several key challenges, including learning and implementing new technologies, addressing performance issues, and managing project scope. Despite these challenges, the team was able to develop a functional and engaging game that can be played in a web browser.

The project's success can be attributed to the dedication and collaboration of the team members, as well as the effective use of version control, risk mitigation strategies, and regular communication. This project not only resulted in a successful final product but also provided valuable learning experiences for the team members in terms of teamwork, project management, and the development of technical skills.

## 8. **BIBLIOGRAPHY**

<https://sketchfab.com/3d-models/shipping-containers-cc3f7136710f4905905eae1d10ac50b7>

<https://sketchfab.com/3d-models/rusty-hatchbacks-low-poly-props-d0d2dda8447a475888c06a6d909c7aa2>

<https://sketchfab.com/3d-models/low-poly-dumpster-093e5b05faa947729b1be24c013e563b>

<https://sketchfab.com/3d-models/beretta-pistol-fps-animation-0313ab1888994c14abeaf444d7af3217>

<https://sketchfab.com/3d-models/makarov-pm-fps-animations-d02ebd58e3cf44acb7af73e3f156be28>

<https://sketchfab.com/3d-models/low-poly-sw-model-39-hushpuppy-b26e4ef77f1547c2b594213e03cf633f>

<https://fertile-soil-productions.itch.io/modular-village-pack>

<https://developer.playcanvas.com/en/tutorials/keyboard-input/#%3Ccode%3Eispressed%3C/code%3E-vs-%3Ccode%3Ewaspressed%3C/code%3E>

<https://developer.playcanvas.com/en/user-manual/scripting/communication/>

<https://developer.playcanvas.com/en/user-manual/user-interface/layout-groups/>

<https://developer.playcanvas.com/en/user-manual/scripting/anatomy/>

<https://developer.playcanvas.com/en/user-manual/scripting/creating-new/>

<https://developer.playcanvas.com/en/user-manual/scripting/script-attributes/>

<https://forum.playcanvas.com/t/using-animations/27114>

<https://developer.playcanvas.com/en/api/pc.AnimComponent.html#speed>

<https://forum.playcanvas.com/t/solved-pause-animation-graph/24291>

<https://forum.playcanvas.com/t/how-to-create-a-custom-function/29095>

<https://developer.playcanvas.com/en/tutorials/mouse-input/>



<https://developer.playcanvas.com/en/user-manual/packs/components/camera/>

<https://developer.playcanvas.com/en/user-manual/graphics/physical-rendering/physical-materials/>

<https://developer.playcanvas.com/en/tutorials/importing-first-model-and-animation/>

<https://developer.playcanvas.com/en/tutorials/ui-elements-leaderboard/>

<https://forum.playcanvas.com/u/tirk182/summary> : A huge thanks to Terry Tirko for all his help during the project, in helping us from very early on.

Thanks also to the playcanvas team, with all its staff for being so responsive in the forums and responding to difficulties that we encountered throughout the production of our game.