**DEERWALK INSTITUTE OF TECHNOLOGY**

**Tribhuvan University**

**Institute of Science and Technology**

# 3D battle royale game

**A PROJECT-II REPORT**

**Submitted to**

**Department of Computer Science and Information Technology**

**DWIT College**

***In partial fulfillment of the requirements for the Bachelor's Degree in Computer Science and Information Technology***

Submitted by

Aarjan Pokharel (802)

Avinawa Acharya (811)

Prashant Neupane (830)

Sagar Giri (836)

17th August, 2022

**DWIT College**

**DEERWALK INSTITUTE OF TECHNOLOGY**

## SUPERVISOR’S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by AARJAN POKHAREL, AVINAWA ACHARYA, PRASHANT NEUPANE AND SAGAR GIRI entitled **“3D BATTLE ROYALE GAME USING UNITY”** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

…………………………………………

Shyam Sundar Khatiwada

Lecturer

DWIT College

Deerwalk Institute of Technology

**DWIT College**

**DEERWALK INSTITUTE OF TECHNOLOGY**

## STUDENT'S DECLARATION

We hereby declare that we are the only author of this work and that no sources other than that listed here have been used in this work.

| | | | |
|---|---|---|---|
| ……………… | ………………… | …………………… | ……………….. |
| Aarjan Pokharel | Avinawa Acharya | Prashant Neupane | Sagar Giri |

17th August, 2022

**DWIT College**

**DEERWALK INSTITUTE OF TECHNOLOGY**

## LETTER OF APPROVAL

This is to certify that this project prepared by AARJAN POKHAREL, AVINAWA ACHARYA, PRASHANT NEUPANE AND SAGAR GIRI entitled **"3D BATTLE ROYALE GAME USING UNITY"** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

| | |
|---|---|
| ……………………………………… | ………………………………………….. |
| Shyam Sundar Khatiwada<br>Lecturer<br>DWIT College | Hitesh Karki<br>Campus Chief<br>DWIT College |

# ACKNOWLEDGEMENT

First and foremost, we would like to express our special gratitude to Mr. Shyam Sundar Khatiwada for helping us to put all our ideas well above the level of simplicity and into something concrete. We got to learn more about game development and working with unity through his motivation. His vision was aspired as well as inspiring, and a corner stone for the progression of this project.

We offer our sincere appreciation for the learning opportunities provided by the Deerwalk Institute of Technology, as well as by Project-II supervisor committee: Mr. Hitesh Karki, Principle; Mr. Bijay Babu Regmi, Professor and HoD; Mr. Niresh Dhakal, Professor and HoD; Mr. Ritu Raj Lamsal, Professor and HoD; Mr. Saurav Gautam, Professor.

Aarjan Pokharel (802)

Avinawa Acharya (811)

Prashant Neupane (830)

Sagar Giri (836)

17th August, 2022

# ABSTRACT

Battle Royale games are multiplayer games where players fight until last man remains. Most of the online battle royale games are designed as first-person shooter which are highly competitive in nature that results in stress and toxicity. Moreover, these games require internet connection and cannot be played in locally hosted network. The proposed game is a 3D multiplayer battle royale game based on the amalgamation of game of tag and bomb blast. The game allows at most 10 players to play at a time. The game does not deal with artificial intelligence and is solely intended for multiplayer use. Both online mode as well as LAN mode is implemented in this game. Unity game engine is used as the game engine and fish net networking is used as network manager to implement multiplayer mode.

For the purpose of enhancing the game play experience, this project includes various abilities. Abilities such as teleport, high jump, etc. adds unique element to the game, which user can exploit to win. To make the game fairer and more competitive coin base system known as luck has been implemented. The luck uses simple easy to understand concept which defines which player will inherit the bomb after each explosion. Game environment follows the symmetrical map pattern, making the game more competitive, less biased and easy to navigate. User with less experience about the battle royale games can also easily understand the game mechanics, map and the game logic. Various environmental obstacles were implemented to increase the player interactivity. Obstacles such as rotating plank created a unique situation were players who missed the timing were punished by being blocked. Features to spectate was implemented, so that the eliminated players can spectate other players and still be part of the game. To enable communication among players, chat feature was implemented. The players can initiate chat with other players and continue to converse until the game is finished.

**Keywords**: *Battle Royale; FPS; LAN; Fishnet; Unity; Multiplayer; C#*

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| 2D | Two Dimensional |
| 3D | Three Dimensional |
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| FPS | First Person Shooter |
| LAN | Local Area Network |
| OOP | Object Oriented Programming |
| PC | Personal Computer |
| PUBG | PlayerUnknown's Battlegrounds |
| UI | User Interface |
| VFX | Visual Eff |

# CHAPTER 1: INTRODUCTION

## 1.1. Overview

Battle royale games has become one of the most popular genres for gaming industry. The idea of battle royale was first introduced in a movie of the same name as genre called “Battle Royale” in 2000. Similarly, this project is a 3D multiplayer battle royale game which is based on the amalgamation of game of tag and bomb blast. Unity game engine is used as main game engine with C# as the OOP language. With this project, we want to give reliable network and game play experience along with ecstasy of nostalgia.

## 1.2. Background and Motivation

Video games are the ultimate form of entertainment. One can interact with the environment and become fully engaged in the narrative. Battle royal mechanics of games such as ‘PUBG’, ‘Fortnite’ and especially a mini section of ‘Crab Game’ were captivating. With the game themed around battle royale, player tend to create their own narrative and tremendously increase the replay ability of the game. Since, in this type of game nothing is predetermined, each match iteration feels special in its own. Unlike other story games where the difficulty setting is fixed, this game the difficulty is based on the actual human performance. With this, there tends to be competitive environment between each and every players. So, a PC game was to be built where player can throw down and compete between friends or random people.

## 1.3. Problem Statement

Most of the battle royale games, only have the concept of first-person shooters. There are no actual games that gives player a dose of nostalgia involving simple childhood games. That’s why the idea to create game based on game of tag and bomb blast was considered.

To play most battle royale games, player need access to the internet, as such when the connectivity is low or the internet is down, player cannot play the game. To address this problem, the game needed to address both LAN mode and online mode.

## 1.4. Objectives

The major objectives of this game project are:

- To create a 3D, game that implements the concept of game of tag and bomb blast.
- To implement actually physics simulation including VFX, particle effects, shaders and animations.
- To implement real time online lobby and game play.
- To implement LAN mode.

## 1.5. Scope

This game is mainly focused for the people who like battle royale, multiplayer party games, casual gamers and also the players who grew up playing traditional game of tag and bomb blast.

## 1.6. Development Methodology

For the development of this project, rapid prototyping development methodology was followed which falls under agile methodology.

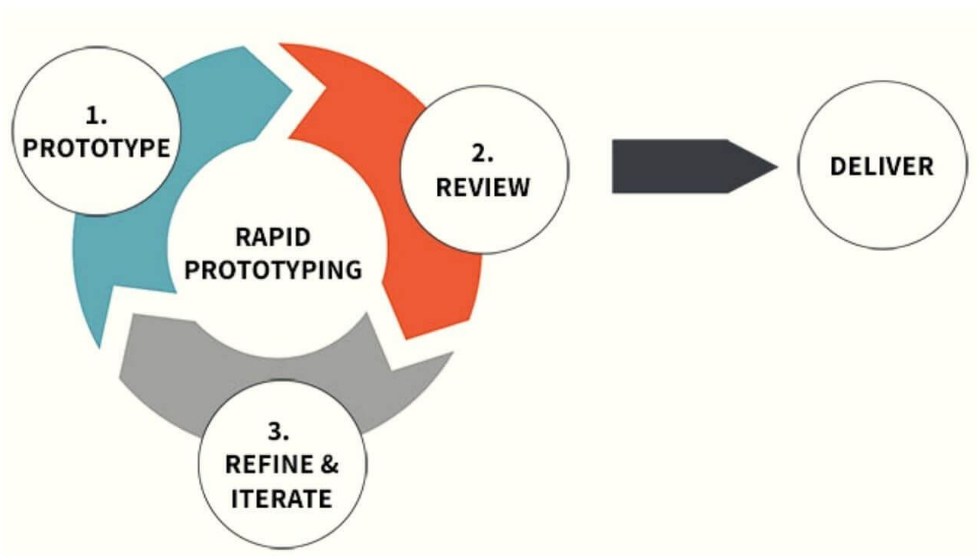


**Figure 1: Rapid Prototyping Development Methodology**

## 1.7. Outline

The outline is divided into various sections with different information presented to the viewer about the report. Various diagrams are also shown as a clear demonstration about the plan, blueprint as well as the output of the project. The outline of the project report is shown below

**Preliminary Section**: Preliminary Section includes various outlines like title page, abstract, acknowledgement, table of contents, list of figures as well as list of tables.

**Introduction Section**: Introduction section consists of outlines which includes the overview of the project, the background and motivation of the project, problem statement with the objectives and scope.

**Requirement and Feasibility Analysis Section**: Requirement and Feasibility Analysis section includes functional and non-functional requirement as well as the research done about the project.

**System Design** Section: This section includes various diagram for understanding the overall design of the game.

**Implementation and Evaluation Section:** This section explains about the creation of the project as well as the evaluation results.

**Conclusion**: This section includes the overall theme and conclusion of the project.

# CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

## 2.1. Background Study

Upon conducting research, it was found that there were very few battle royale games in the early 2010s but after the introduction of PUBG, battle royale games exploded in popularity. There was a total of 20 million unique players playing a battle royale game within a timeframe of 24 hours [1]. Choosing the right game engine, in terms of revenue, technology and efforts – can be a crucial step to consider for the success of any gaming project.

## 2.2. Literature Review

Until recently, the cost of licensing one of the premier game engines has ranged from several hundred thousand to several million dollars per title, relegating the dream of creating one's own 3D game to an unattainable fantasy. With Unity's bold move to offer a robustly featured free version of their engine, a radical change in the pricing models of the high-end engines has rocked the industry which be willing to take high cost to make games or CG (Computer Graphics) [2].Unity 3D game engine is the most professional, steady and efficient game engine, and Unity 3D game engine supports Web, PC, Mac, iOS, Flash, Android, Xbox360, PS3 and Wii platforms [3, 4]. The project used the Unity 3D game engine to develop a 3D game, which the case of the game is a FPS game. Unity 3D pack includes a free version, has rich asset Store, supports Cross Platform and has easy development of Multiplayer games. Fish-Networking assets have proven to perform 4 times faster as compared to Mirror Networking and uses little as 13% the bandwidth of Mirror's internal components [5].

## 2.3. Current System

Currently, there are some similar games which depicts the concept of multiplayer games such as Bomb Squad, Bomber Friend, and Bomber man [6].

## 2.4. The problem with Current System

However, these games are nearly impossible to enjoy due to insane number of ads. Also, most of these games are 2D which are not as immersive and realistic as compared to the concept of a 3D game. Furthermore, the current games only work via internet connection. In a country like Nepal with unstable and unreliable internet service, it gives us a frustrating gameplay experience.

This game project tries to solve the advertisement problem by giving users an ad free experience. Moreover, the internet requirement issue could be fixed by providing LAN feature where users could host and play games locally with their friends.

# CHAPTER 3: SYSTEM ANALYSIS

## 3.1. Requirement Analysis

The requirements analysis is broken down into two sections namely function requirements and non-functional requirements and they are discussed below.

### 3.1.1. Functional Requirement

- The user shall be able to join online lobby.
- The user shall be able to interact with game world in real time.
- The user shall be able to have in game chat.
- The user shall be able to play in first person perspective.
- The user shall be able to use various power spawned throughout the map.
- The user shall be able to perform various game mechanics like running, throwing, jumping and teleporting.
- The user shall be able to view their match history.
- The user shall be able to view the players in the current lobby.
- The user shall be able to interact with coins in the game environment.

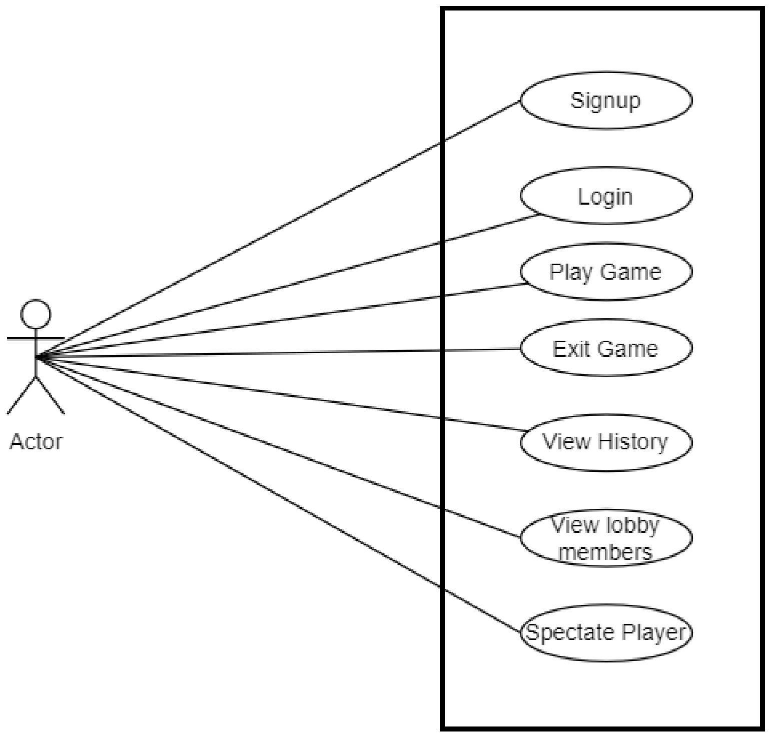


**Figure 2: Use case diagram**

### 3.1.2. Non-Functional Requirement

- The connection and service must be reliable.
- The service must be easy to use.
- It must be cheap to use the service.
- The game must have interactive and user-friendly user interface.
- The game must show the difference between attacker and survivor.
- The game must reveal the location of player categorized as "survivor" to the player categorized as "Attacker"
- The game must swap the category for attacker and survivor when the attacker successfully attacks the survivor.
- The game must show the acquired coins for each player.

## 3.2. Feasibility Analysis

### 3.2.1. Technical Feasibility

The technical feasibility of this project is divided into 2 sub section of software and hardware. In the software section following software were used.

- Unity game engine for gameplay and physics simulation
- Fish net network solution
- Mixamo for character animation
- Blender for modeling
- Photoshop for UI and textures.

And in hardware section, personal laptop and AWS server to host network solution were used. Hence, the project was technically feasible.

### 3.2.2. Operational Feasibility

As listed in the requirement analysis section, this project will implement all the features mentioned. The networking solution is based on client server architecture with the addition of server authority and host migration. Which will enable multiplayer game play with in cloud as well as LAN network. Since, the network solution is authoritative in nature, it will prevent the plague of hacking present in other multiplayer games. Along with these, the game play will focus on low poly cell shaded environment, it will run on low end to high end hardware. Hence, the project is operationally feasible.

### 3.2.3. Economic Feasibility

The resources that were used in this project were mostly free and was made by the team. The LAN game with friends will also not require any additional resources. Hence, this project is economically feasible in nature.

### 3.2.4. Schedule Feasibility

A Gantt chart is used to do the timetable feasibility study. A list was made of all the critical tasks for this project and a timeline was prepared indicating that all of them could be accomplished by the deadline.

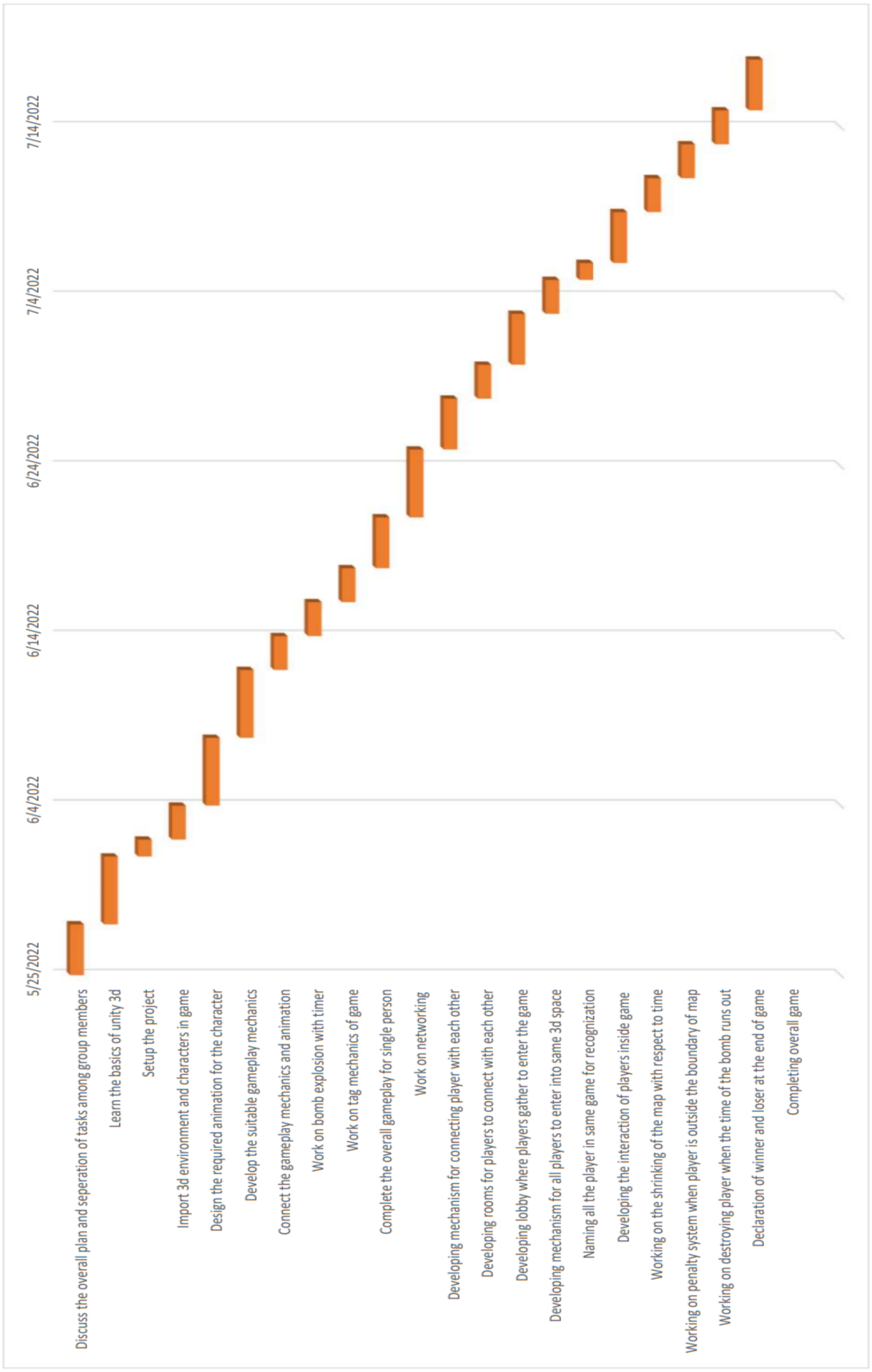


**Figure 3: Gantt Chart**

## 3.3. Analysis

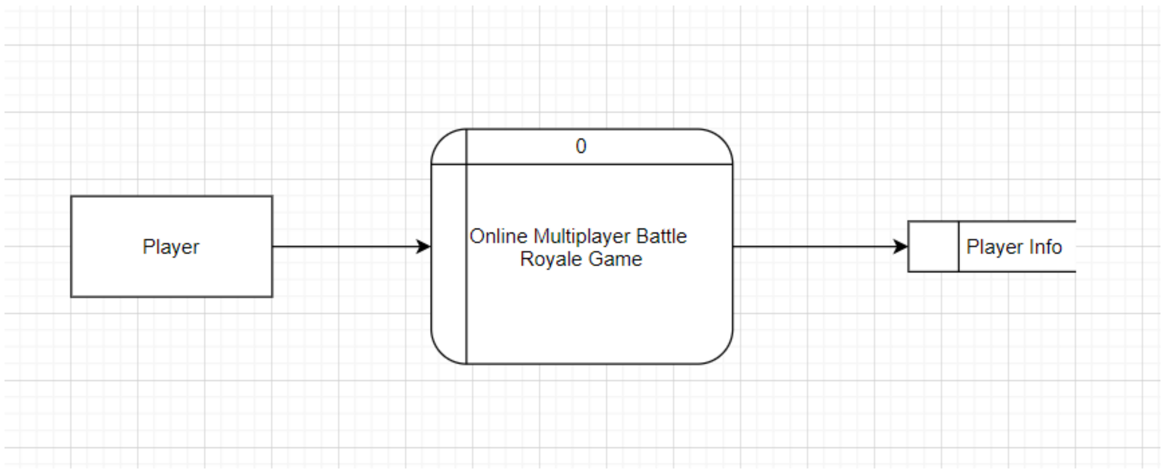


**Figure 4: Context Free Diagram**

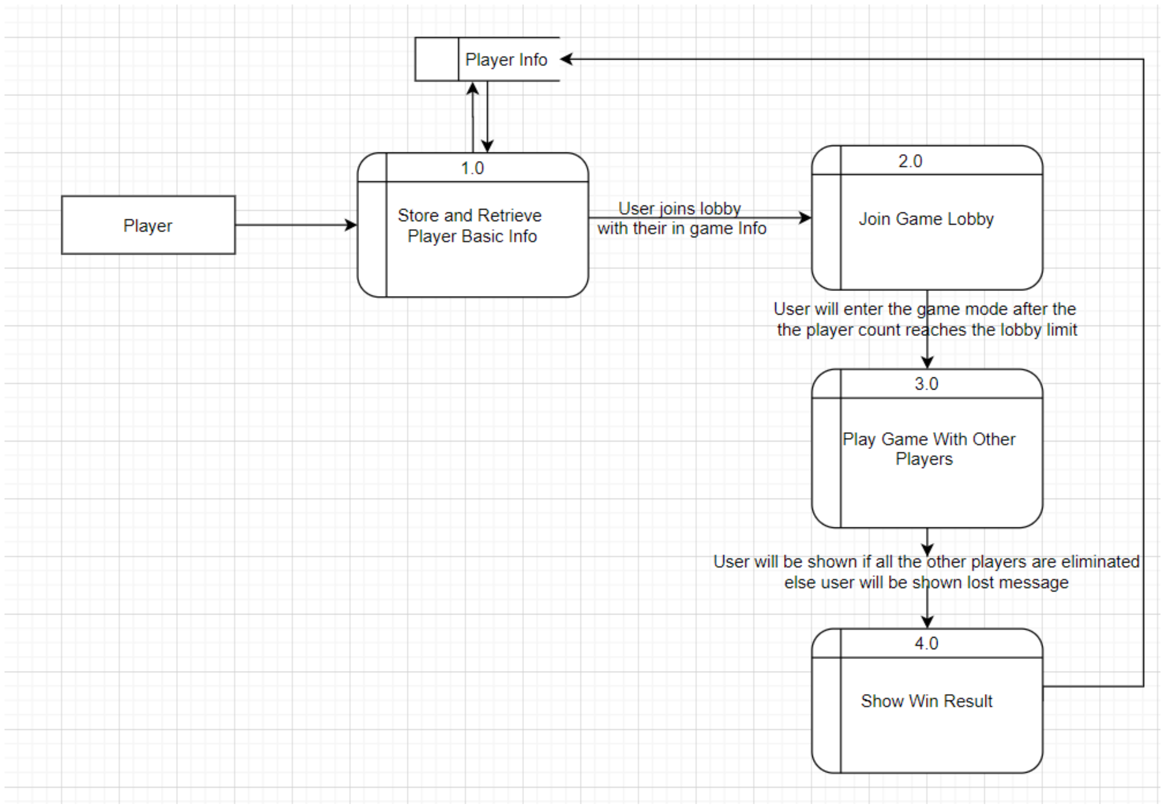

Player Info
1.0
Store and Retrieve Player Basic Info
Player
User joins lobby with their in game Info
2.0
Join Game Lobby
User will enter the game mode after the the player count reaches the lobby limit
3.0
Play Game With Other Players
User will be shown if all the other players are eliminated else user will be shown lost message
4.0
Show Win Result

**Figure 5: Data Flow Diagram**

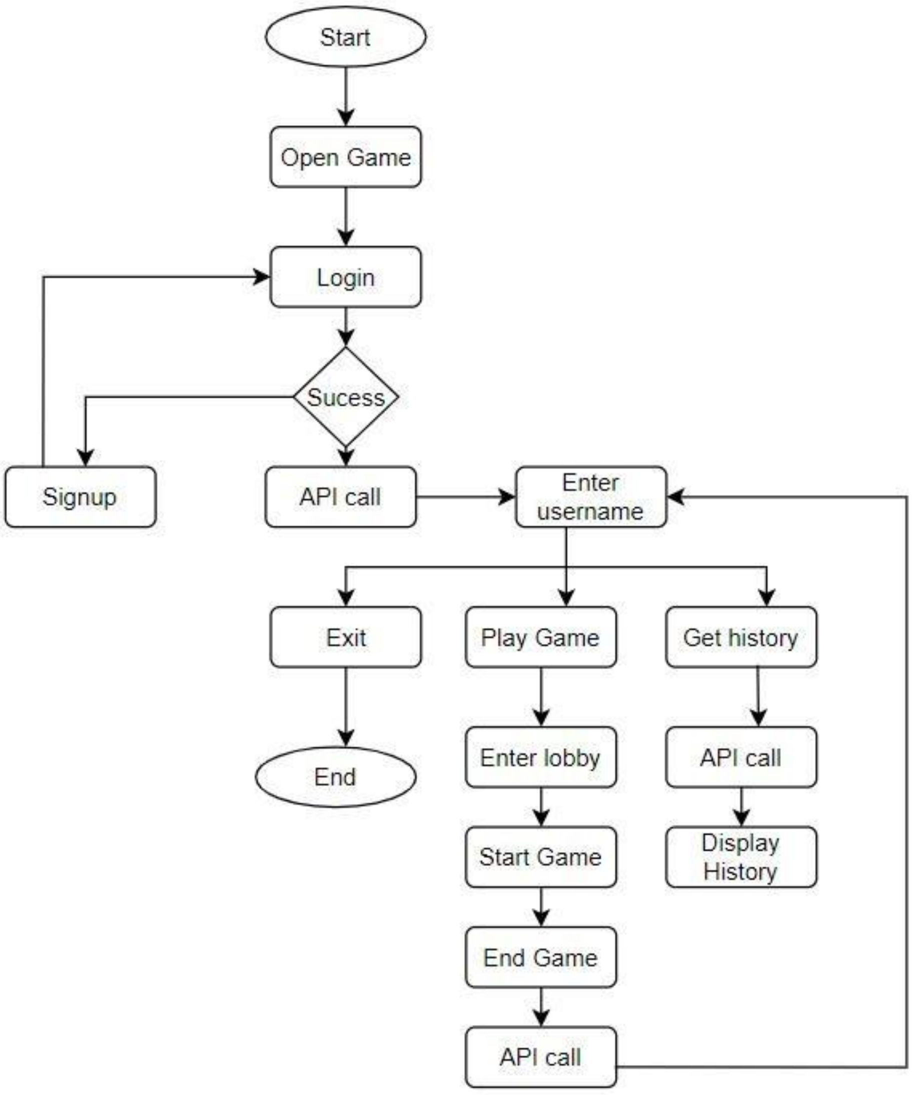


**Figure 6: Flow chart diagram**

# CHAPTER 4: SYSTEM DESIGN

## 4.1. Design

### 4.1.1. System Architecture

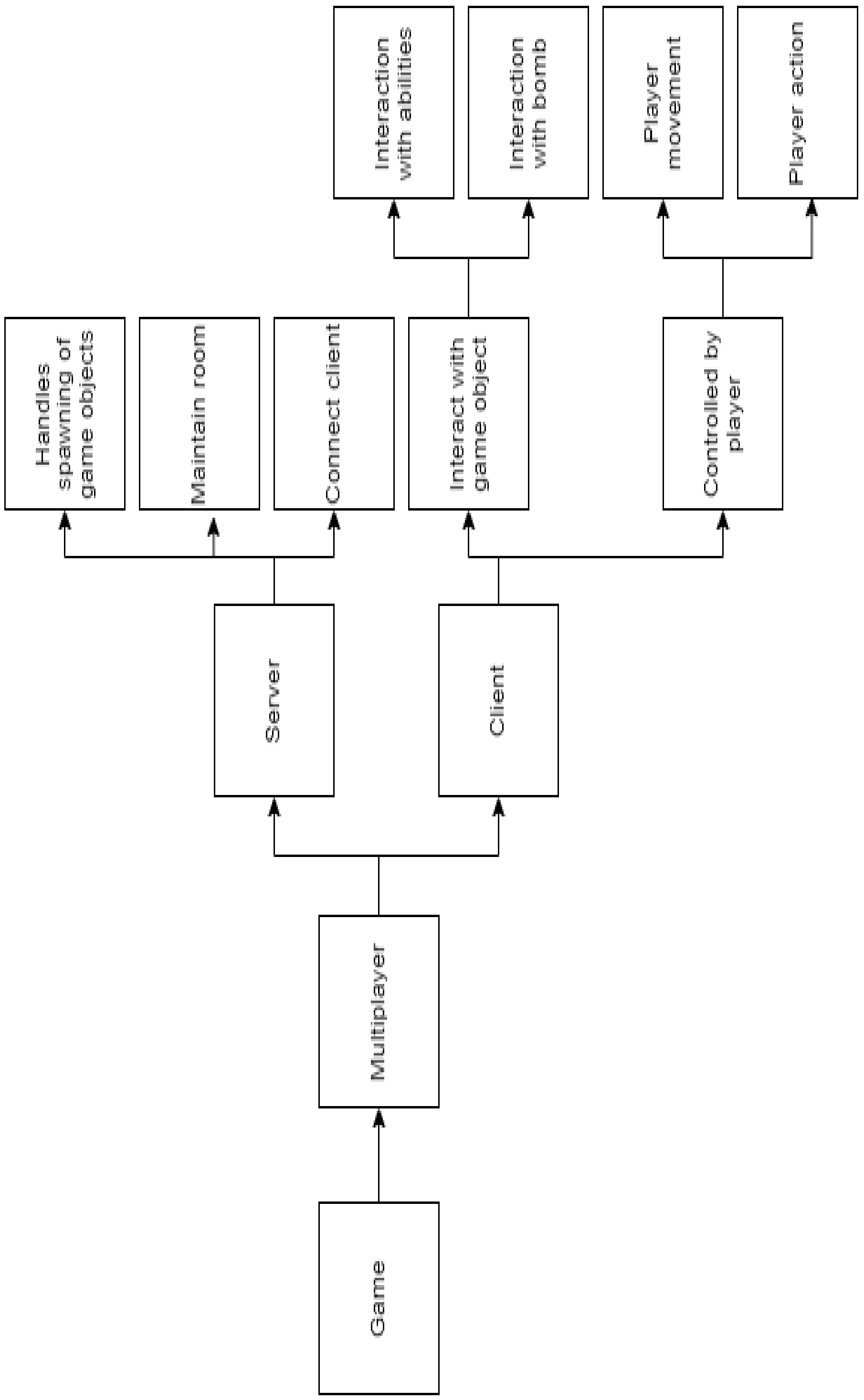


Figure 7: System Architecture

## 4.2. Algorithm Details

### Bcrypt

The problems present in traditional UNIX password hashes led naturally to a new password scheme which is called bcrypt, referring to the Blowfish encryption algorithm. Bcrypt uses a 128-bit salt and encrypts a 192-bit magic value. It takes advantage of the expensive key setup in eksblowfish.

The bcrypt algorithm runs in two phases, in the first phase, EksBlowfishSetup is called with the cost, the salt, and the password, to initialize eksblowfish's state. Most of bcrypt's time is spent in the expensive key schedule. Following that, the 192-bit value “OrpheanBeholderScryDoubt" is encrypted 64 times using eksblowfish in ECB mode with the state from the previous phase. The output is the cost and 128-bit salt concatenated with the result of the encryption loop.

Bcrypt algorithm was used in the project to perform encryption in the saved password using node js.

### JWT Authentication

A JSON web token (JWT) is JSON Object which is used to securely transfer information over the web (between two parties). It can be used for an authentication system and can also be used for information exchange. The token is mainly composed of header, payload, and signature. [4]JSON Web Tokens are used in the industry more and more. The spec which defines them (RFC7519) describes them as a compact, URL-safe means of representing claims between parties by encoding them as JSON objects which can be digitally signed or encrypted.

JWT algorithm was used to construct access tokens for authenticating users.

# CHAPTER 5: IMPLEMENTATION AND TESTING

## 5.1. Implementation

The login page will be presented to users once they have completed the registration process. The system can be used by users who have registered and users can also play as a guest. After that, users can choose their in-game name, the name can be edited before any game. After entering the preferred name, users can get access to the main menu. The main menu provides users with the options to play the game, check their game history, logout from their account or exit the game. Upon selecting to play, users have the option to either host the game or be the client on the host IP. After that, the players get joined into the game lobby until a maximum of 10 players is reached or if the host decides to start the game. Then, the players are spawned in the game environment and the game is started with one of the players given a bomb randomly. The "bomber" tries to pass the bomb to the "runners" while the "runners" try to run away from the "bomber". Some abilities are spawned all over the game map which the players can access giving them special abilities such as teleportation, high jump and high speed for a certain amount of time. The map is filled with "luck coins" that can be interacted by all players that determines who gets the bomb after a player has died. The bomb goes off every 35 seconds, thus a person is eliminated from the game every 35 seconds after which they can either spectate the alive players or exit to main menu.

All the user interfaces were firstly designed in Photoshop whereas all the game objects were modeled, textured and animated using Blender. These models were then imported and integrated using Unity's own editor.

The problems encountered during implementing the system were:

1. Version Control using Git required Git Premium version for the projects network size.
2. The system could not be tested for online system as port forwarding was not allowed by the available ISPs.
3. The hosting sites for Node JS like Heroku, vercel are quite slow and take some time for handling API requests.

4. The game could not be implemented in mobile version given the time frame of the project.

### 5.1.1. Tools Used

- Unity game engine:

  Unity game engine provides tools such as rendering, shading, physics, animator, etc. to create game. It also provides cross platform services to create game for multiple different platforms. Unity was used to create gameplay mechanics and game environment.

- Fish net network solution:

  Fish net networking solution provides networking capability such as hosting, remote procedure calls, syncing, etc. Fish net was used to host the game, provide a connection for users to join and sync changes among players.

- Mixamo:

  Mixamo is a library of thousands of character animations captured from professional motion actor using motion capture suit. Mixamo was used for animation of third person character.

- Blender:

  Blender is a free and open-source 3D creation suite which supports entirety of 3D pipeline such as modelling, rigging, animation, texturing, shading, rendering, etc. Blender was used to model all of the 3D game objects in the game.

- Photoshop:

  Photoshop is an image creation, graphic design and photo editing software designed by Adobe. Photoshop was used to create various textures for 3D models and images for UI.

- Node JS:

  Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on a JavaScript Engine (i.e. V8 engine) and executes

JavaScript code outside a web browser, which was designed to build scalable network applications. It was used to develop and deploy various API used in the game.

- C# :
  C# is a general-purpose, multi-paradigm programming language. It is used as official scripting language for unity. C# was used to script various game behavior.

- Mongo DB:
  MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. Mongo DB was used to store users information such as login credentials and game history.

- Postman:
  Postman is an API client that makes it easy for developers to create, share, test and document APIs. This is done by allowing users to create and save simple and complex HTTP/s requests, as well as read their responses. Postman was used for API testing.

- GitHub:
  GitHub is an internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project. GitHub was used as a version control for the project.

### 5.1.2. Implementation Details of Modules

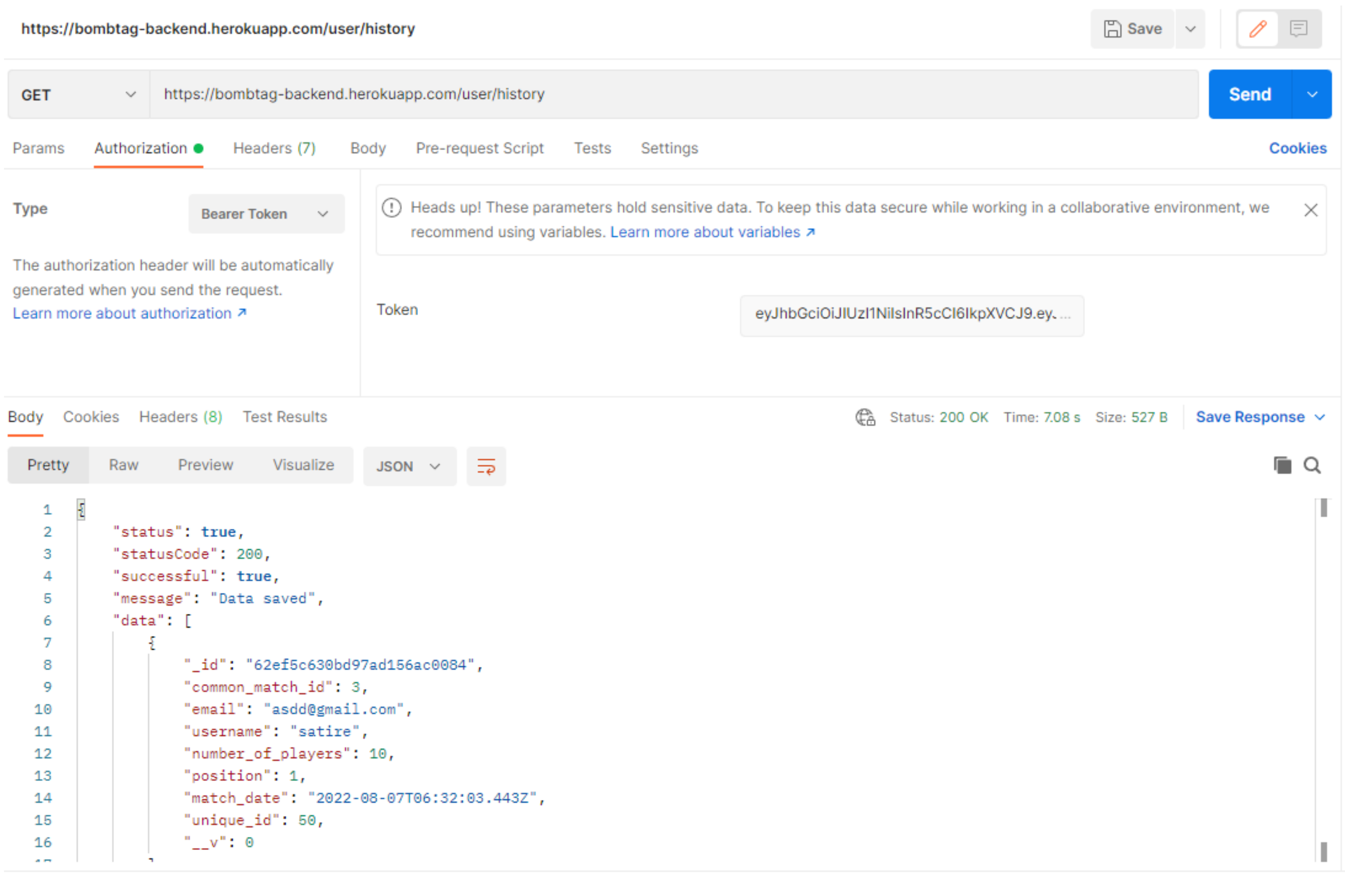
The game construction can be described in the following major steps:

1. The User Interface
2. Backend and API integration
3. Integrating Game Objects
4. Constructing game logic and interaction with environment

5. Audio Module
6. Multiplayer system and control logic
7. Chat Module

The user interface was created using Unity's built-in engine. The backend logic was created was created using Node JS. All the player information was stored in MongoDB Atlas using Node JS. Various endpoints were created for handling different routes. The API contained 4 endpoints:

1. /sendplayerdata: Validates and stores users signup data.
2. /user/login: Implements authentication system.
3. /user/history: Checks JWT and returns Match History.
4. /user/sendhistory: Saves current Match Statistics.


https://bombtag-backend.herokuapp.com/user/history

GET https://bombtag-backend.herokuapp.com/user/history

Authorization

Type: Bearer Token

The authorization header will be automatically generated when you send the request.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables.

Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...

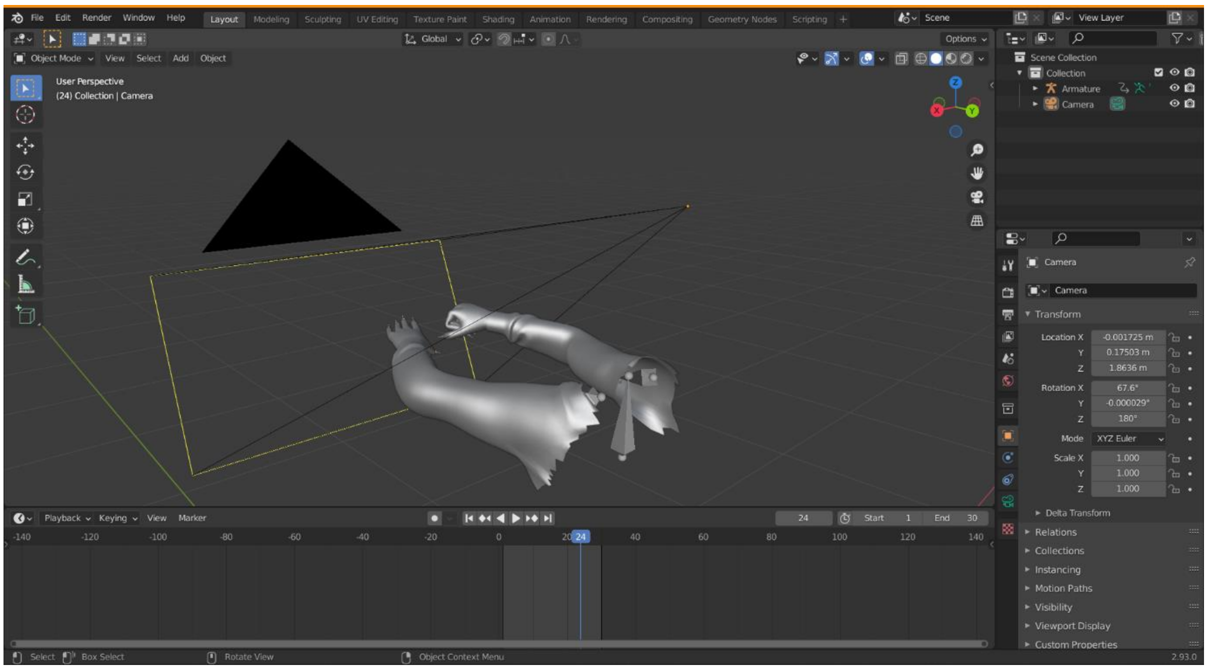
Status: 200 OK Time: 7.08 s Size: 527 B

```
{
    "status": true,
    "statusCode": 200,
    "successful": true,
    "message": "Data saved",
    "data": [
        {
            "_id": "62ef5c630bd97ad156ac0084",
            "common_match_id": 3,
            "email": "asdd@gmail.com",
            "username": "satire",
            "number_of_players": 10,
            "position": 1,
            "match_date": "2022-08-07T06:32:03.443Z",
            "unique_id": 50,
            "__v": 0
```

**Figure 8: Sending API request using postman**

The models were created using Blender by using its mechanisms such as:

- Modeling
- Animating
- Texturing
- Baking
- UV editing
- PNG rendering, etc.



Camera

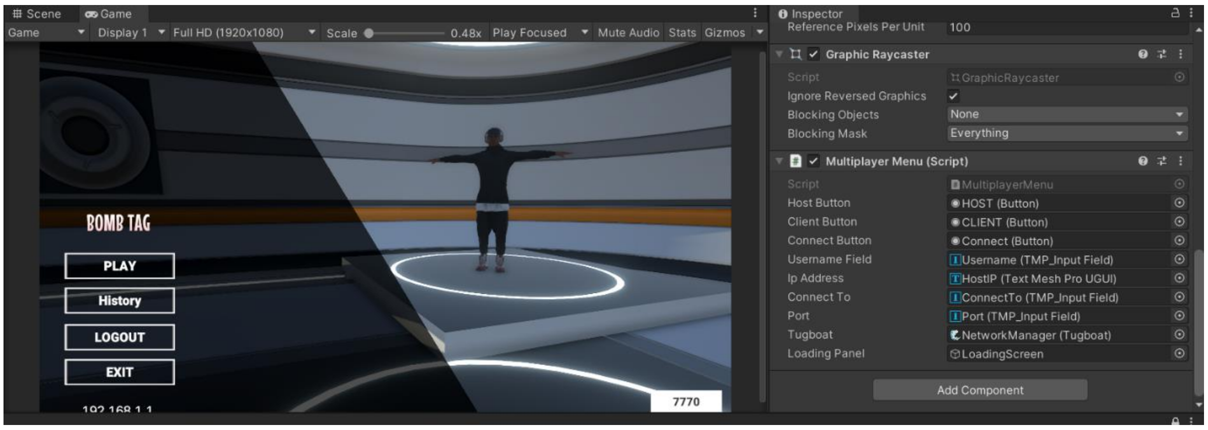
Transform

Location X -0.001725 m
Y 0.17503 m
Z 1.8636 m
Rotation X 67.6°
Y -0.000029°
Z 180°
Mode XYZ Euler
Scale X 1.000
Y 1.000
Z 1.000

Delta Transform
Relations
Collections
Instancing
Motion Paths
Visibility
Viewport Display
Custom Properties

**Figure 9: Modelling player hand using blender**

The core game mechanism such as running, jumping, teleporting, etc. was introduced using Player Character Controller with Unity’s own Physics System. All the game logics including players details, players death, API call, etc. was handled using C#. The game objects that were imported using Blender were integrated with their corresponding scripts. The scripts included different object aspect such as object collider, animator controller, 3d object physics, object initialization and object mechanism.

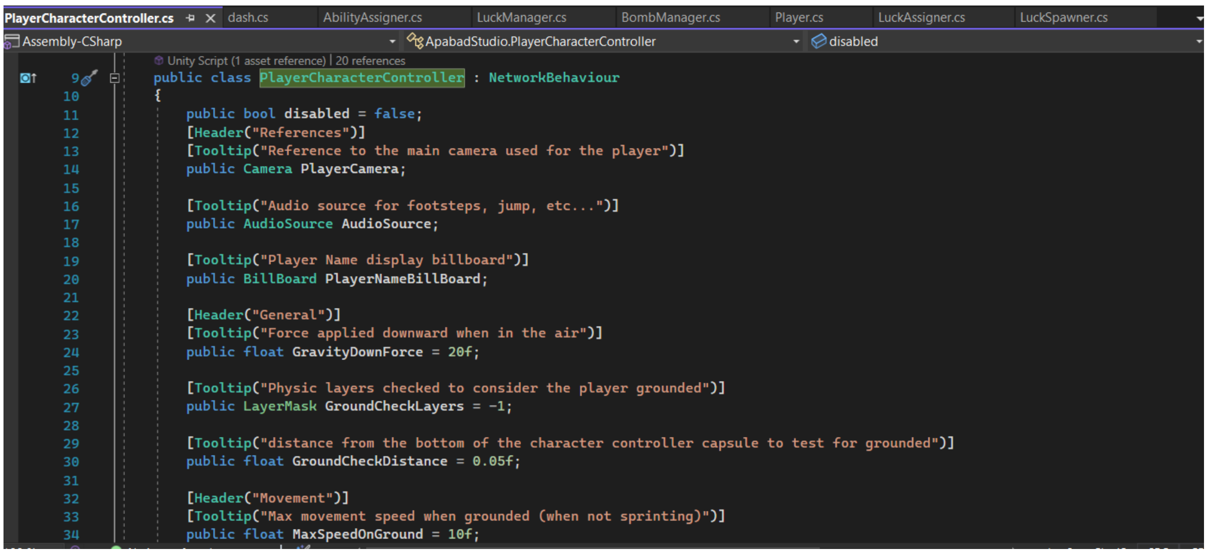


Reference Pixels Per Unit 100

Graphic Raycaster
Script GraphicRaycaster
Ignore Reversed Graphics ✓
Blocking Objects None
Blocking Mask Everything

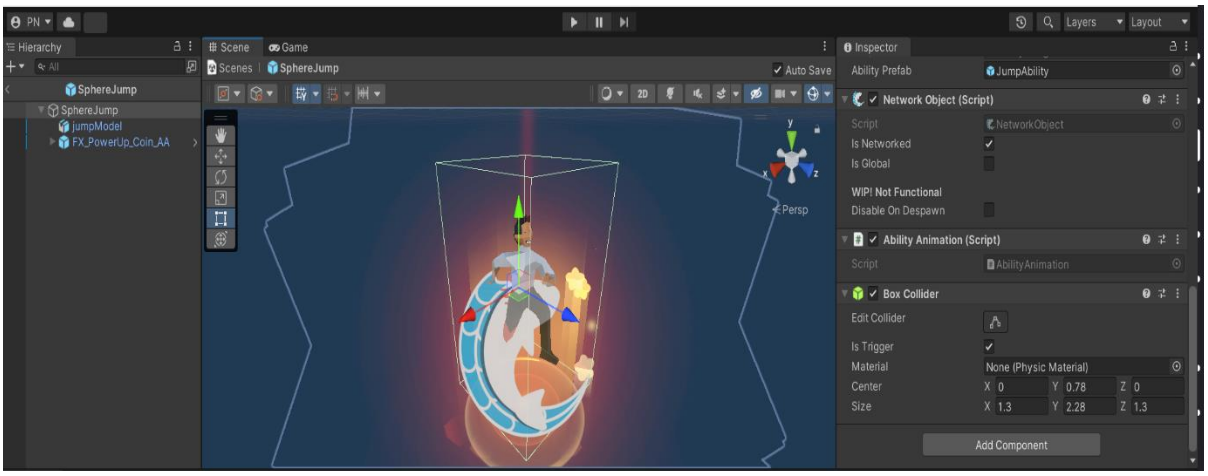
Multiplayer Menu (Script)
Script MultiplayerMenu
Host Button HOST (Button)
Client Button CLIENT (Button)
Connect Button Connect (Button)
Username Field Username (TMP_Input Field)
Ip Address HostIP (Text Mesh Pro UGUI)
Connect To ConnectTo (TMP_Input Field)
Port Port (TMP_Input Field)
Tugboat NetworkManager (Tugboat)
Loading Panel LoadingScreen

Figure 10: Objects integrated with scripts


PlayerCharacterController.cs
dash.cs
AbilityAssigner.cs
LuckManager.cs
BombManager.cs
Player.cs
LuckAssigner.cs
LuckSpawner.cs
Assembly-CSharp
ApabadStudio.PlayerCharacterController
disabled
Unity Script (1 asset reference) | 20 references

```
public class PlayerCharacterController : NetworkBehaviour
{
    public bool disabled = false;
    [Header("References")]
    [Tooltip("Reference to the main camera used for the player")]
    public Camera PlayerCamera;

    [Tooltip("Audio source for footsteps, jump, etc...")]
    public AudioSource AudioSource;

    [Tooltip("Player Name display billboard")]
    public BillBoard PlayerNameBillBoard;

    [Header("General")]
    [Tooltip("Force applied downward when in the air")]
    public float GravityDownForce = 20f;

    [Tooltip("Physic layers checked to consider the player grounded")]
    public LayerMask GroundCheckLayers = -1;

    [Tooltip("distance from the bottom of the character controller capsule to test for grounded")]
    public float GroundCheckDistance = 0.05f;

    [Header("Movement")]
    [Tooltip("Max movement speed when grounded (when not sprinting)")]
    public float MaxSpeedOnGround = 10f;
```

Figure 11: Player controller class



Ability Prefab: JumpAbility
Network Object (Script)
Script: NetworkObject
Is Networked: ✓
Is Global:
WIP! Not Functional
Disable On Despawn:
Ability Animation (Script)
Script: AbilityAnimation
Box Collider
Edit Collider
Is Trigger: ✓
Material: None (Physic Material)
Center: X 0 Y 0.78 Z 0
Size: X 1.3 Y 2.28 Z 1.3

**Figure 12: Particle effect using unity editor**

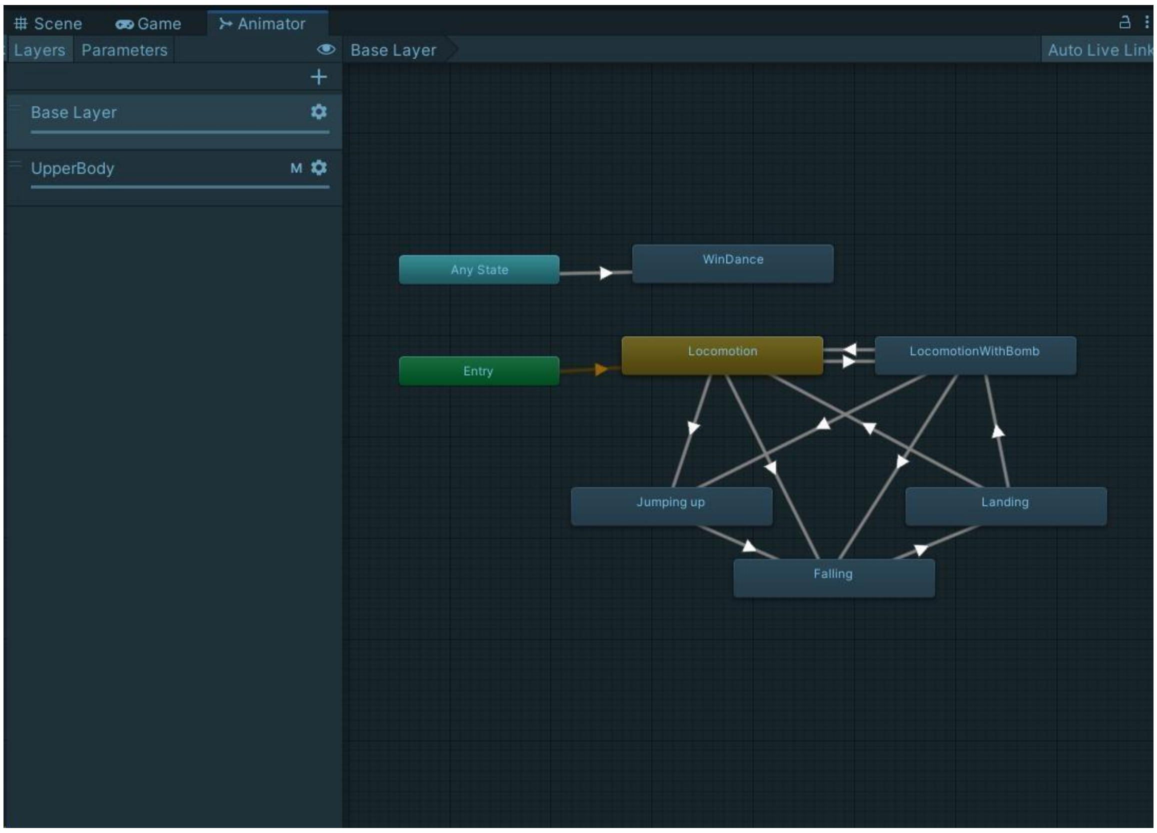
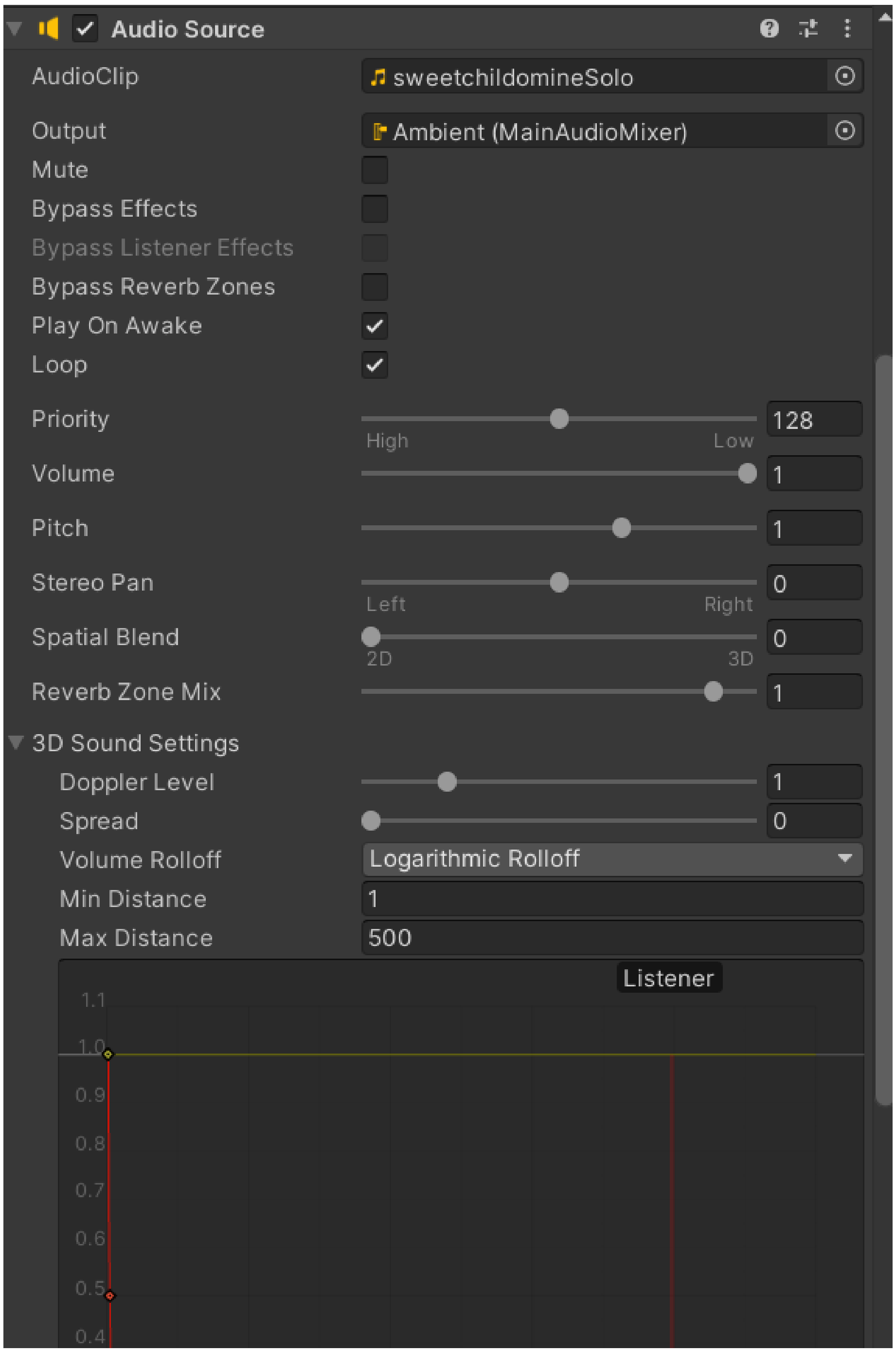


**Figure 13: Animator to control animations using unity editor**

The music for game was collected from various artists. Similarly, the sound effects were mainly sourced from freesound.org. The sounds were played using the AudioSource component of unity game engine.


Audio Source

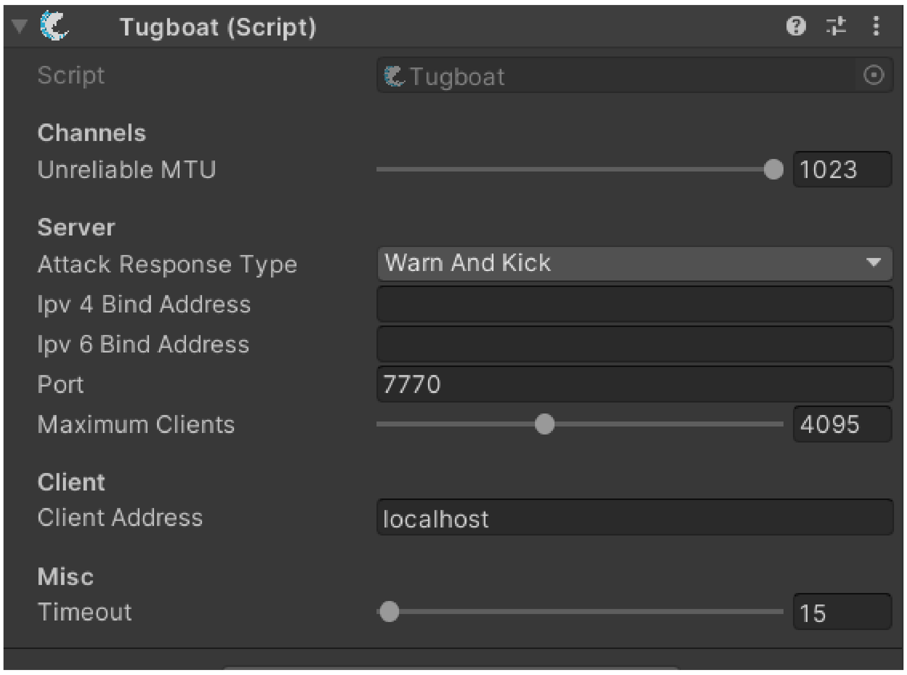
| | |
|---|---|
| AudioClip | sweetchildomineSolo |
| Output | Ambient (MainAudioMixer) |
| Mute | ☐ |
| Bypass Effects | ☐ |
| Bypass Listener Effects | ☐ |
| Bypass Reverb Zones | ☐ |
| Play On Awake | ✓ |
| Loop | ✓ |
| Priority | 128 |
| Volume | 1 |
| Pitch | 1 |
| Stereo Pan | 0 |
| Spatial Blend | 0 |
| Reverb Zone Mix | 1 |
| 3D Sound Settings | |
| Doppler Level | 1 |
| Spread | 0 |
| Volume Rolloff | Logarithmic Rolloff |
| Min Distance | 1 |
| Max Distance | 500 |

**Figure 14: AudioSource Component integration**

To implement networking in the game fishnet's tugboat was used. Tugboat uses LiteNetLib to support reliable and unreliable messages. This is the default transport for Fish-Networking. It provides component settings such as

- Unreliable MTU is the maximum size a packet may be on the unreliable channel.
- Port indicates which port the server will listen on.

- Maximum Clients is how many players can be connected to the server at once.
- Client Address is the IP of the server, which clients connect to. Localhost is the default value for local testing.
- Timeout is how long in seconds the transport must go without data before disconnecting.

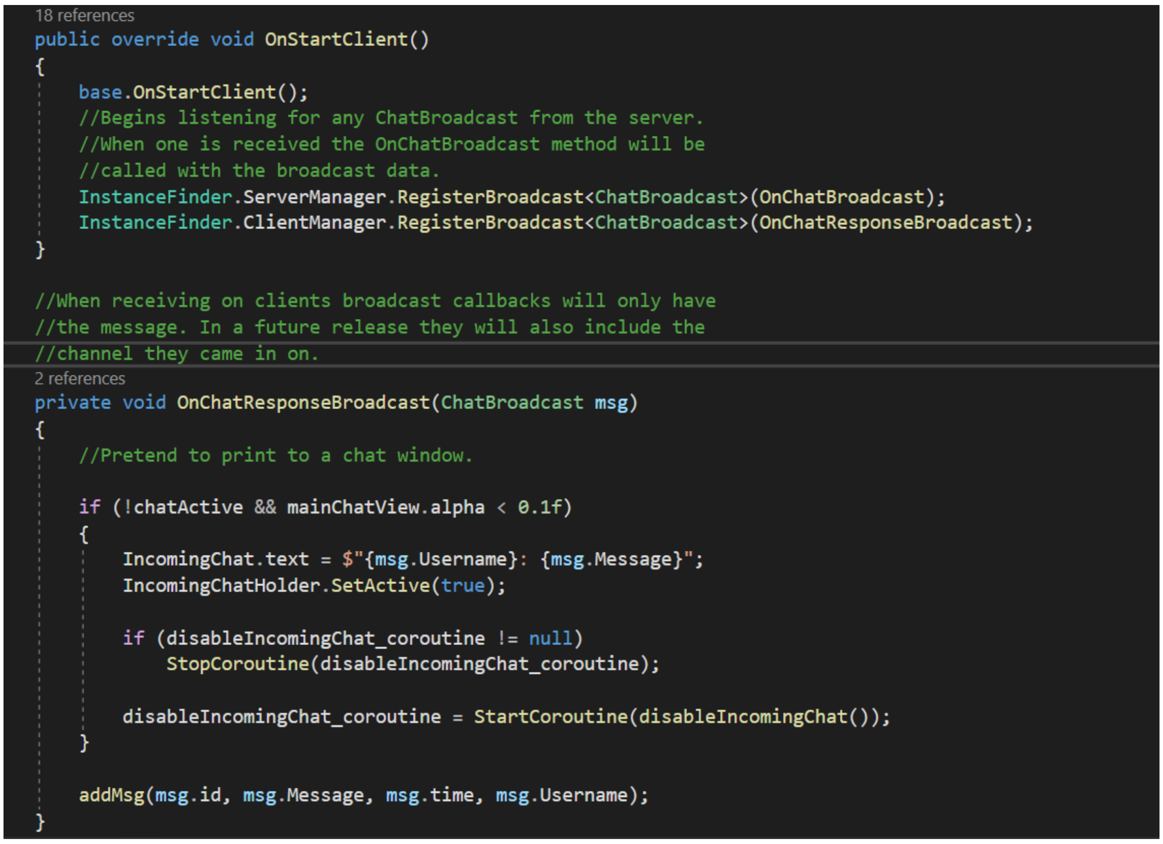


Tugboat (Script)

| | |
|---|---|
| Script | Tugboat |
| **Channels** | |
| Unreliable MTU | 1023 |
| **Server** | |
| Attack Response Type | Warn And Kick |
| Ipv 4 Bind Address | |
| Ipv 6 Bind Address | |
| Port | 7770 |
| Maximum Clients | 4095 |
| **Client** | |
| Client Address | localhost |
| **Misc** | |
| Timeout | 15 |

**Figure 15: Network Transport Module**

In order to communicate between players, real time chat was implemented using fishnets broadcast functionality. The registered users can send and receive instant chat messages from other connected players.



```
18 references
public override void OnStartClient()
{
    base.OnStartClient();
    //Begins listening for any ChatBroadcast from the server.
    //When one is received the OnChatBroadcast method will be
    //called with the broadcast data.
    InstanceFinder.ServerManager.RegisterBroadcast<ChatBroadcast>(OnChatBroadcast);
    InstanceFinder.ClientManager.RegisterBroadcast<ChatBroadcast>(OnChatResponseBroadcast);
}

//When receiving on clients broadcast callbacks will only have
//the message. In a future release they will also include the
//channel they came in on.
2 references
private void OnChatResponseBroadcast(ChatBroadcast msg)
{
    //Pretend to print to a chat window.

    if (!chatActive && mainChatView.alpha < 0.1f)
    {
        IncomingChat.text = $"{msg.Username}: {msg.Message}";
        IncomingChatHolder.SetActive(true);

        if (disableIncomingChat_coroutine != null)
            StopCoroutine(disableIncomingChat_coroutine);

        disableIncomingChat_coroutine = StartCoroutine(disableIncomingChat());
    }

    addMsg(msg.id, msg.Message, msg.time, msg.Username);
}
```

**Figure 16: Chat Implementation Using Broadcast**

## 5.2. Testing

### 5.2.1. Test Cases for Unit Testing

The following are the components that were tested for Unit Testing:

**Table 1: Unit testing test cases for sign up form**

| Test Case | Description | Test Data | Expected Result | Actual Result | Remarks |
|---|---|---|---|---|---|
| TU01 | When the player enters correct name, email, username and password for registering. | Name = “Aarjan Pokhrael”<br>Email = “abc@gmail.com”<br>Username = “lonewolf”<br>Password = “@Test123”<br>Confirm password = “@Test123” | Player should be registered to the game. | As expected | PASS |
| TU02 | When the player enters incorrect username while registering. | Name = “Aarjan Pokhrael”<br>Email = “abc@gmail.com”<br>Username = “”<br>Password = “@Test123”<br>Confirm password = “@Test123” | The game must show the invalid username error. | As expected | PASS |
| TU03 | When the player enters incorrect email while registering. | Name = “Aarjan Pokhrael”<br>Email = “abc”<br>Username = “Lonewolf”<br>Password = “@Test123” | The game must show invalid email error | Failed as incorrect regex was used for validating email. | FAIL |

| | | | | | |
|---|---|---|---|---|---|
| | | Confirm password = "@Test123" | | | |
| TU04 | When player enters incorrect password while registering. | Name = "Aarjan Pokhrael"<br>Email = "abc@gmail.com"<br>Username = "Lonewolf"<br>Password = "123"<br>Confirm password = "@Test123" | The game must show incorrect password error. | As expected | PASS |
| TU05 | When player enters incorrect name while registering. | Name = "Aarjan"<br>Email = "abc@gmail.com"<br>Username = "Lonewolf"<br>Password = "@Test123"<br>Confirm password = "@Test123" | The game must show invalid name error. | Failed as incorrect regex was used for validating full name. | FAIL |
| TU06 | When a player enters different password while confirming password. | Name = "Aarjan"<br>Email = "abc@gmail.com"<br>Username = "Lonewolf"<br>Password = "@Test123"<br>Confirm password = "@Test123" | The game must show invalid password error | As expected. | PASS |
| TU07 | When the player enters incorrect email while registering. | Name = "Aarjan Pokhrael"<br>Email = "abc"<br>Username = "Lonewolf"<br>Password = "@Test123" | The game must show invalid email error | As expected. | PASS |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  |  | Confirm password = "@Test123" |  |  |  |
| TU08 | When player enters incorrect name while registering. | Name = "Aarjan"<br>Email = "abc@gmail.com"<br>Username = "Lonewolf"<br>Password = "@Test123"<br>Confirm password = "@Test123" | The game must show invalid name error. | As expected. | PASS |

**Table 2: Unit testing test cases for login form**

| Test Case | Test Case Description | Test Data | Expected Result | Actual Result | Pass /Fail |
|---|---|---|---|---|---|
| TU01 | Fill in email and password as login details and click login button. | email = “abc@gmail.com” password=”@Test123” | User should be logged in to the game account | As Expected | Pass |
| TU02 | Fill in email and password as login details and click login button. | email = “prashant@gmail.com” password=” Top!gun2” | User should be logged in to the game account | As Expected | Pass |
| TU03 | Click the login button without filling in login details | email = “ ” password = “ ” | User should be shown validation error | As Expected | Pass |
| TU04 | Fill in email and password as login details and click login button. | email = “ abc ” password=“q%3gCM” | User should be logged in to the game account. | User was not found | Fail |
| TU05 | Fill in email and password as login details and click login button. | email = “giree@gmail.com ” password = “ aSDFq” | User should be logged in to the game account. | Failed due to invalid password | Fail |
| TU06 | Fill in email and password as login details and click login button. | email = “nishan@gmail.com” password = “A12345-z” | User should be logged in to the game account. | Failed due to unregistered user | Fail |

**Test for API URL:** <host>/user/history

**Table 3: Unit testing test cases for authentication API using JWT**

| Test Case | Test Case Description | Test Data | Expected Result | Actual Result | Remarks |
|---|---|---|---|---|---|
| TU01 | Providing bearer token in order to fetch user history | Header:{ ‘Authorization’:’bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InNhdGlyZSIsImVtYWlsIjoiYXNkZEBnbWFpbC5jb20iLCJpYXQiOjE2NTk4NTM3MzR9.Vv2LyQHKMF1TKS0WI20XwGkDQmH1TsmAkWRBY9DJEiQ’} | User’s game history should be fetched | As Expected | Pass |
| TU02 | Providing invalid bearer token in order to fetch user history | Header:{ ‘Authorization’:’bearer NcRfTjWnZr4u7x!A%D*G-KaPdSgVkXp2s5v8y/B?E(H+MbQeThWmZq3t6w9z$C&F)J@NcRfUjXn2r5u7x!A%D*GKaPdSgVkYp3s6v9y/B?E(H+MbQeThWmZq4t7w!z%’ } | User’s game history should be fetched | As Expected | Pass |
| TU03 | Not providing any authentication token | Header{} | User’s game history should be fetched. | User was able to fetch history data because authorization was not | Fail |

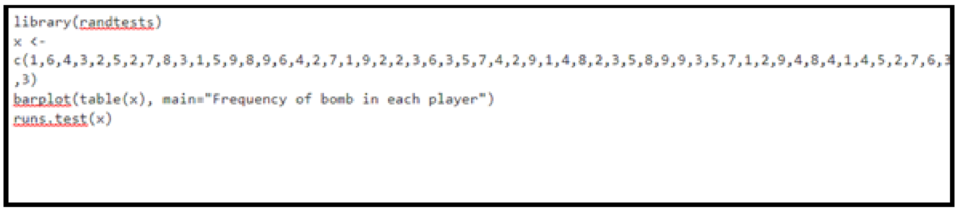
|  |  |  |  |  |  |
|---|---|---|---|---|---|
|  |  |  |  | checked in node JS |  |
| TU04 | Not providing any authentication token | Header{} | User’s game history should be fetched. | As Expected | Pass |

### 5.2.2 Test Cases for System Testing

**Table 4: System testing test cases**

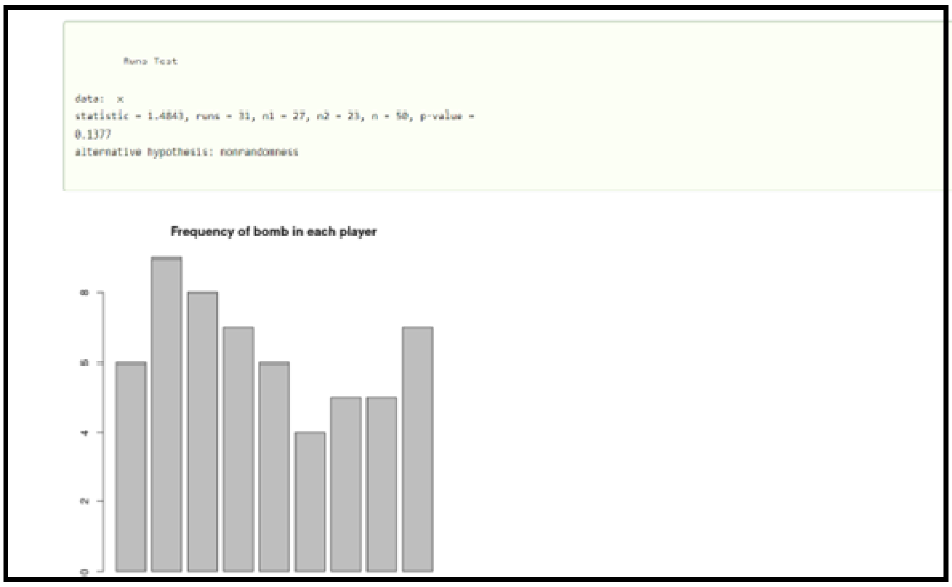
| Test Case | Description | Expected Result | Actual Result | Remarks |
|---|---|---|---|---|
| TU01 | When a player dies and bomb is passed to another player | The player with minimum luck should have the bomb | The bomb was passed to the player with minimum luck | PASS |
| TU02 | When the player throws the bomb and it does not interact with another player | The bomb must return to the player within 4 seconds | The bomb returned to the player after 15 seconds | FAIL |
| TU03 | When the player throws the bomb and it interacts with another player. | The bomb must be passed to another player within 2 seconds | The bomb was passed to another player within 2 seconds" | PASS |
| TU04 | When the bomb is initially spawned in the players hand | The spawning of the bomb must be done random | The bomb was spawned randomly | PASS- Tested using Run Test |
| TU05 | When the game ends | The last standing player must be declared the winner | The last standing player was declared the winner | PASS |
| TU06 | When a player sends message through in-game chat | The message must be sent in real time | The message was sent in real time | PASS |
| TU07 | When the bomb timer ends | The player currently holding the bomb must be eliminated | The player was eliminated | PASS |

| | | | | |
|---|---|---|---|---|
| TU08 | When the host starts the game | Every player in the game lobby must be loaded into the main game environment | Every player was loaded into the main game environment | PASS |
| TU09 | When the client connects to the correct host IP address | The client should be taken to the lobby | The client was taken to the lobby | PASS |
| TU10 | When the players accesses the match history | The match history must be displayed | The match history was displayed | PASS |
| TU11 | When the player throws the bomb and it does not interact with another player | The bomb must return to the player within 4 seconds | The bomb returned to the player within 4 seconds | PASS (Resolving issue of test case 2) |



```
library(randtests)
x <-
c(1,6,4,3,2,5,2,7,8,3,1,5,9,8,9,6,4,2,7,1,9,2,2,3,6,3,5,7,4,2,9,1,4,8,2,3,5,8,9,9,3,5,7,1,2,9,4,8,4,1,4,5,2,7,6,
,3)
barplot(table(x), main="Frequency of bomb in each player")
runs.test(x)
```

**Figure 17: R-Code for testing randomness using Wald–Wolfowitz run test**



```
        Runs Test

data:  x
statistic = 1.4843, runs = 31, n1 = 27, n2 = 23, n = 50, p-value =
0.1377
alternative hypothesis: nonrandomness
```

**Figure 18: Output showing random results of bomb spawn**

## 5.3 Result Analysis

As mentioned above, different white box testing and black box testing were conducted during the testing phase to check the system workings and performance. The black box testing was performed for form validation, login and sign up and gameplay mechanism. The white box testing was performed to test the API calls and their expected output. Regression testing was conducted throughout the development process by the programmer while adding new components to the system.

**Algorithms used for testing randomness:**

Wald–Wolfowitz run test was used to test the randomness of bomb spawn.

1. Formulate the null and alternate hypothesis.

   $H_{null}$: The sequence was produced in a random manner

   $H_{alt}$: The sequence was not produced in a random manner

2. Calculate the test statistic, Z as :

   $Z = (R – R')/S_R$

Where,

R = The number of observed runs

R' = The number of expected runs, given as:

R’ = ((2*n1*n2)/(n1+n2))+1

$S_R$ = Standard Deviation of the number of runs

$S^2_{R\,=}$ (2*n1*n2 (2*n1*n2-n1-n2)) / $(n1+n2)^2$ (n1+n2-1)

With n1 and n2 = the number of positive and negative values in the series

3. Compare the value of the calculated Z-statistic with $Z_{critical}$ for a given level of confidence ($Z_{critical}$ =1.96 for confidence level of 95%). The null hypothesis is rejected i.e. the numbers are declared not to be random, if $|Z| > Z_{critical}$

# CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATION

## 6.1. Conclusion

This project was done to overcome the problem of lack of local multiplayer 3D battle royale games. The aim of creating this game was to provide entertainment and fun atmosphere by allowing the users to play with their friends even in a place where Internet may not be available.

The players can play the game online or locally and also have the ability to view their recent matches. Players can easily join a common lobby by connecting to a single hotspot. Thus a fun party game was created by infusing the concept of Bomb Blast and Game of Tag.

## 6.2. Future Recommendation

The game is currently available only on Windows PCs. The plan for the future of the game is to expand its platform including android and IOS devices, thus making it a cross platform game. Similarly, only a single map is available to play at currently and the future work includes adding other maps into the game. Moreover, currently the game only allows players to join locally but global implementation can be added in the future.

# REFERENCES

[1] "Steam Charts," Steam, 20 04 2022. [Online]. Available: https://steamdb.info/graph/. [Accessed 25 04 2022].

[2] 17 07 2019. [Online]. Available: https://dcubetechnologies.com/casestudies/deep-analysis-and-benefits-of-unity-game-engine-with-comparison-to-other-game-development-frameworks/. [Accessed 25 04 2022].

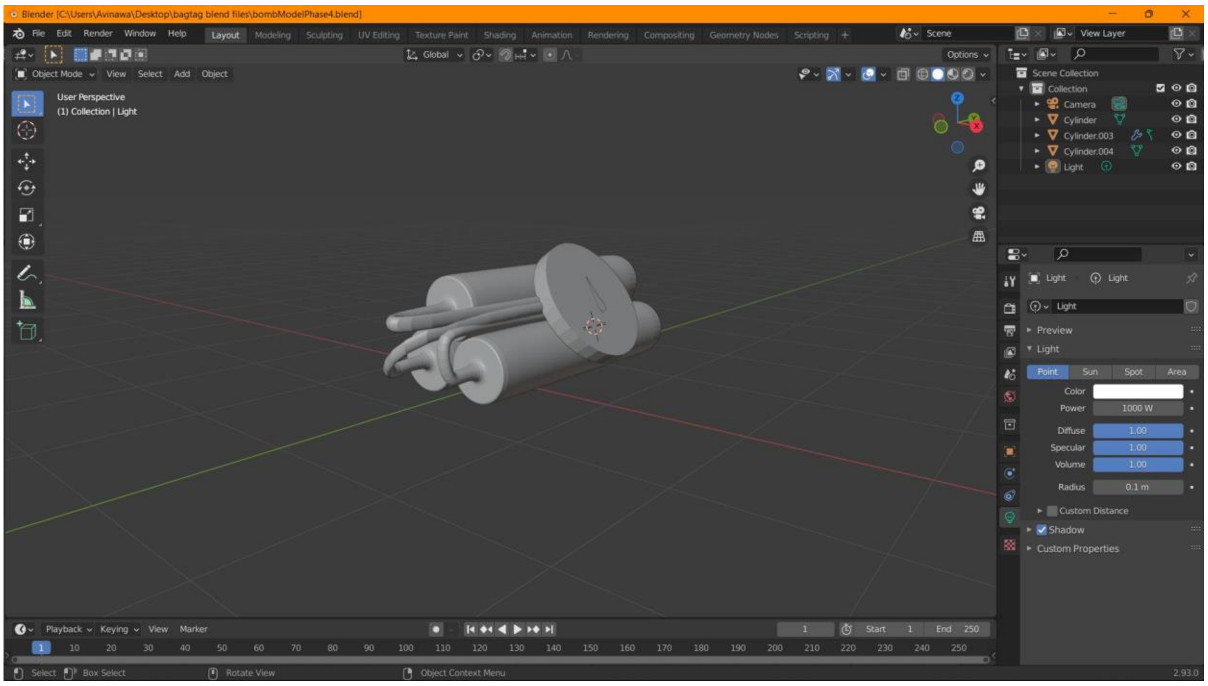
[3] A. K. Peters, "An All-in-One Guide to Implementing Game Mechanics, Art, Design and Programming," in *Holistic Game Development with Unity*, CRC Press, 2017, p. 472.

[4] "auth0," 17 12 2019. [Online]. Available: https://auth0.com/blog/json-web-token-signing-algorithms-overview/. [Accessed 25 07 2022].

[5] "Fish-Net: Networking Evolved," GitBook, 20 01 2022. [Online]. Available: https://fish-networking.gitbook.io/docs/manual/general/performance. [Accessed 28 04 2022].

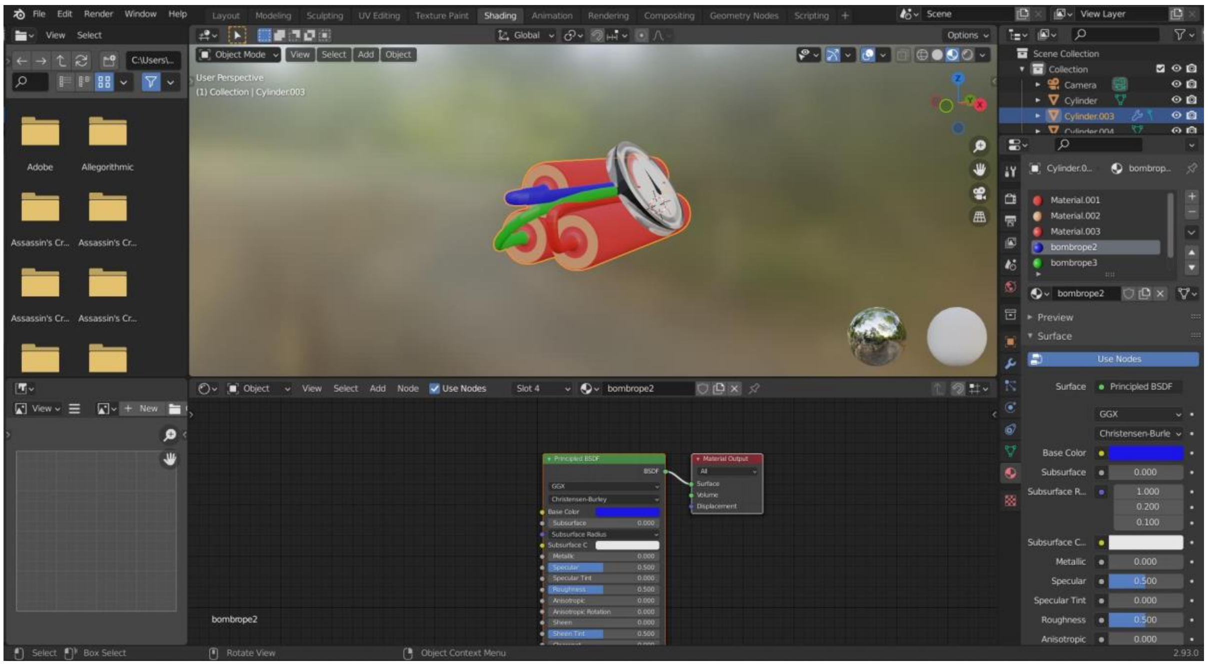
[6] [Online]. Available: “Bomb games - games with a Bang,” agame.com. [Online]. Available: https://www.agame.com/games/bomb-games. [Accessed: 15-Aug-2022]. .

# APPENDIX


Blender [C:\Users\Avinawa\Desktop\bagtag blend files\bombModelPhase4.blend]
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
User Perspective
(1) Collection | Light
Scene Collection
Collection
Camera
Cylinder
Cylinder.003
Cylinder.004
Light

Point
Color
Power 1000 W
Diffuse 1.00
Specular 1.00
Volume 1.00
Radius 0.1 m
Custom Distance
Shadow
Custom Properties

Figure i: 3D modelling for bomb


File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
User Perspective
(1) Collection | Cylinder.003
Material.001
Material.002
Material.003
bombrope2
bombrope3
Principled BSDF
Material Output
bombrope2

Use Nodes
Surface Principled BSDF
GGX
Christensen-Burley
Base Color
Subsurface 0.000
Subsurface R... 1.000 0.200 0.100
Subsurface C...
Metallic 0.000
Specular 0.500
Specular Tint 0.000
Roughness 0.500
Anisotropic 0.000

Figure ii: Texturing bomb model

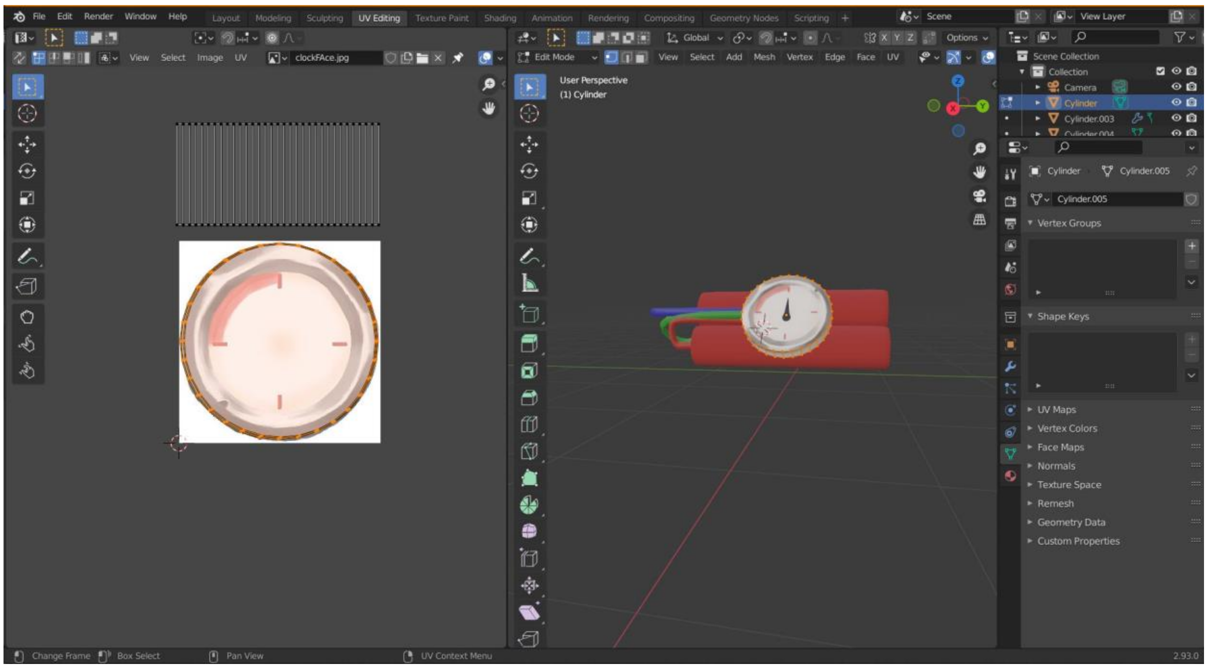


**Figure iii: UV editing bomb model**

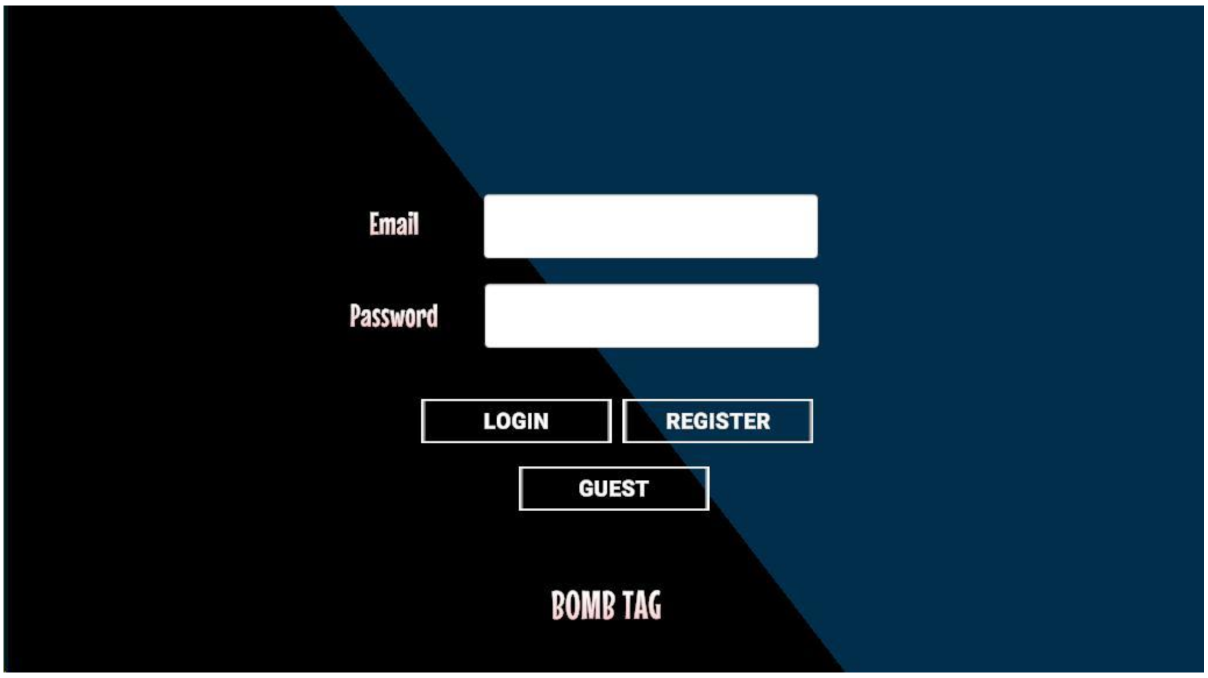


**Figure iv: Login scene**

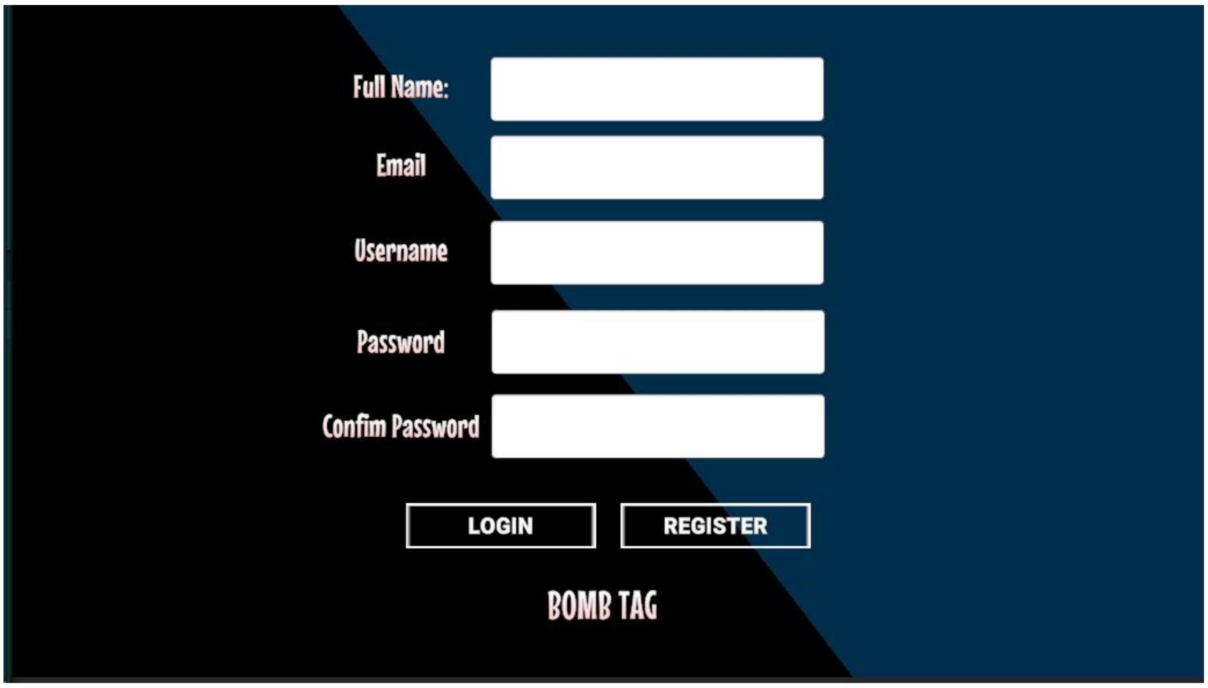


**Figure v: Register scene**

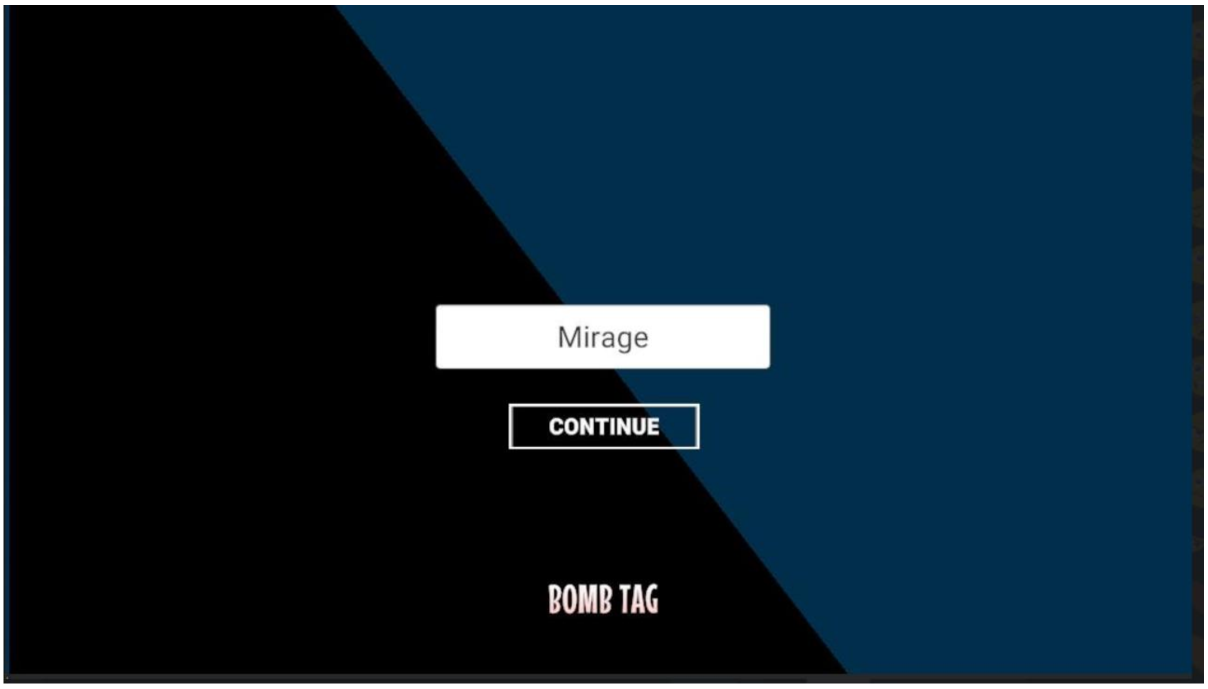


**Figure vi: Adding username to the player**

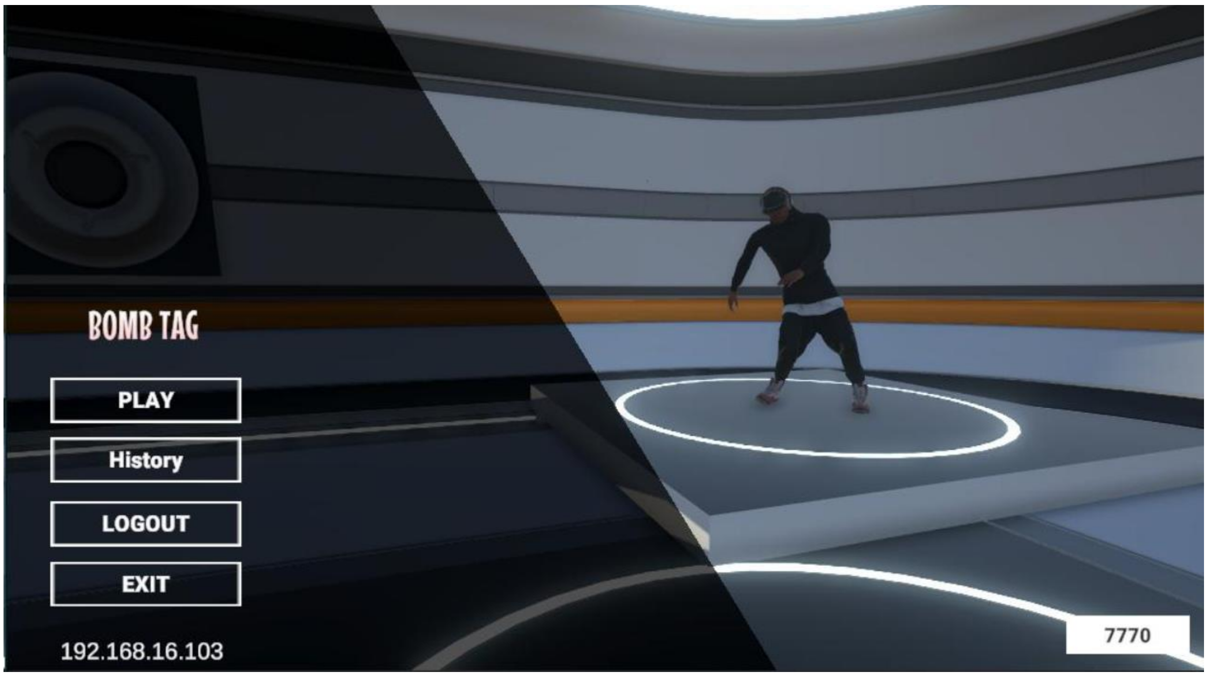


**Figure vii : Main menu**

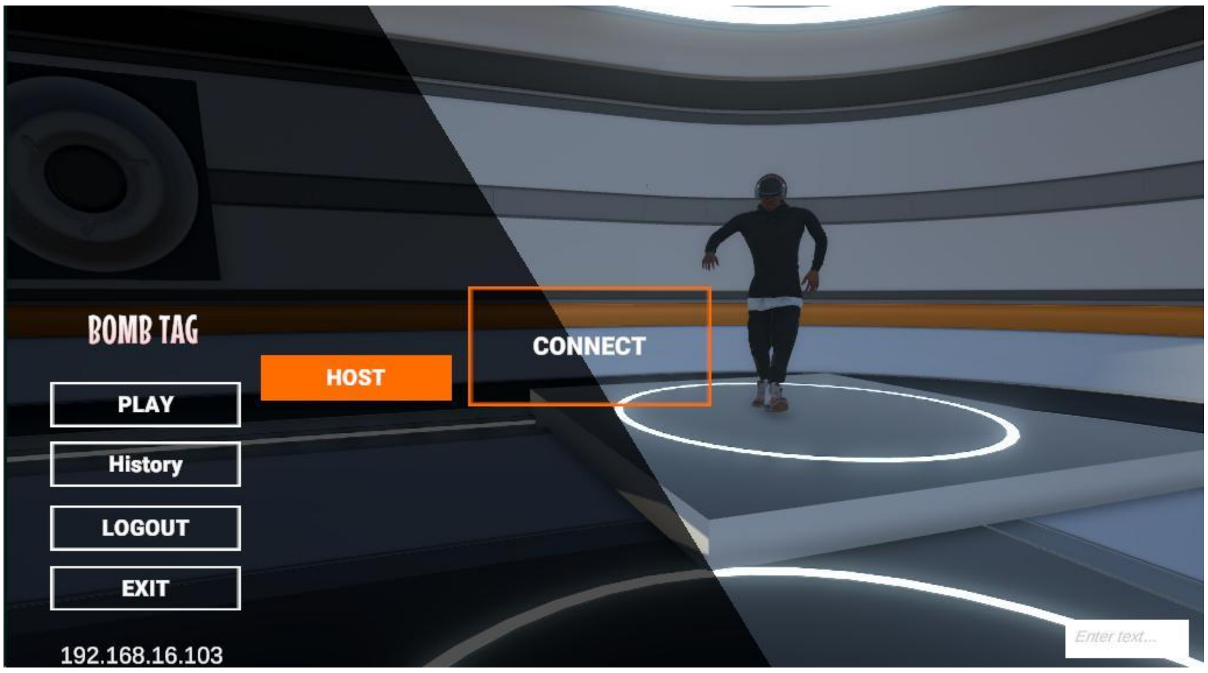


**Figure viii: Hosting the game**

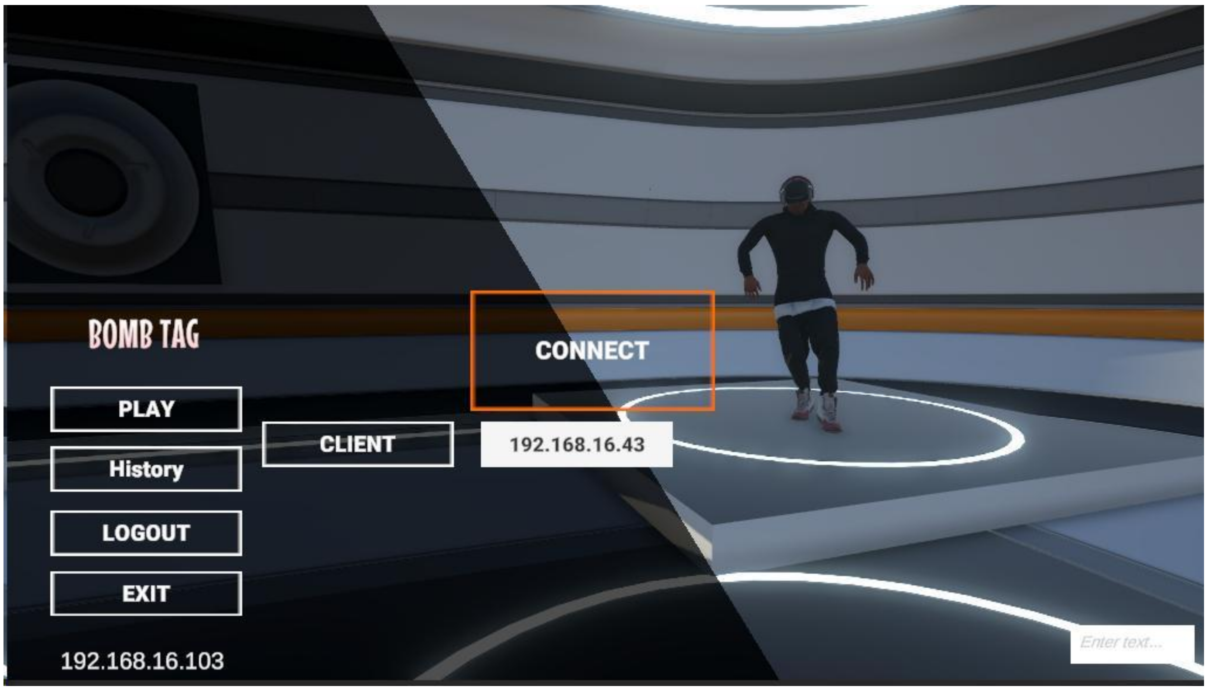


**Figure ix: Client connecting to host game**

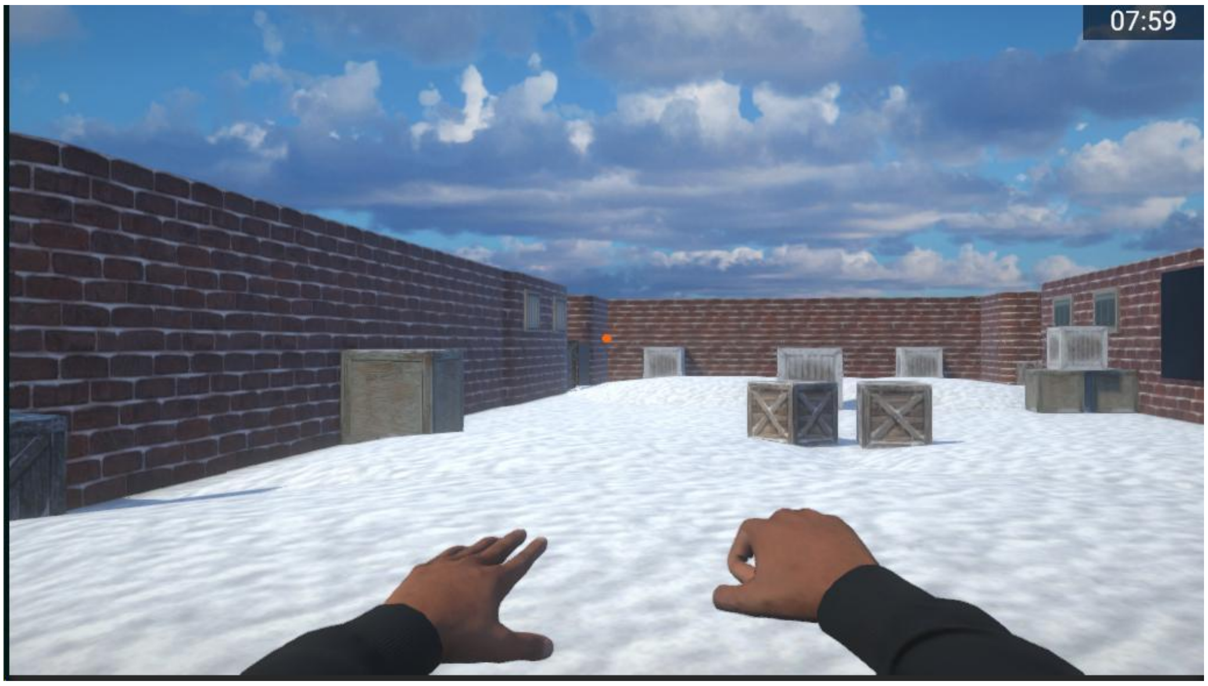


**Figure x: Lobby scene**

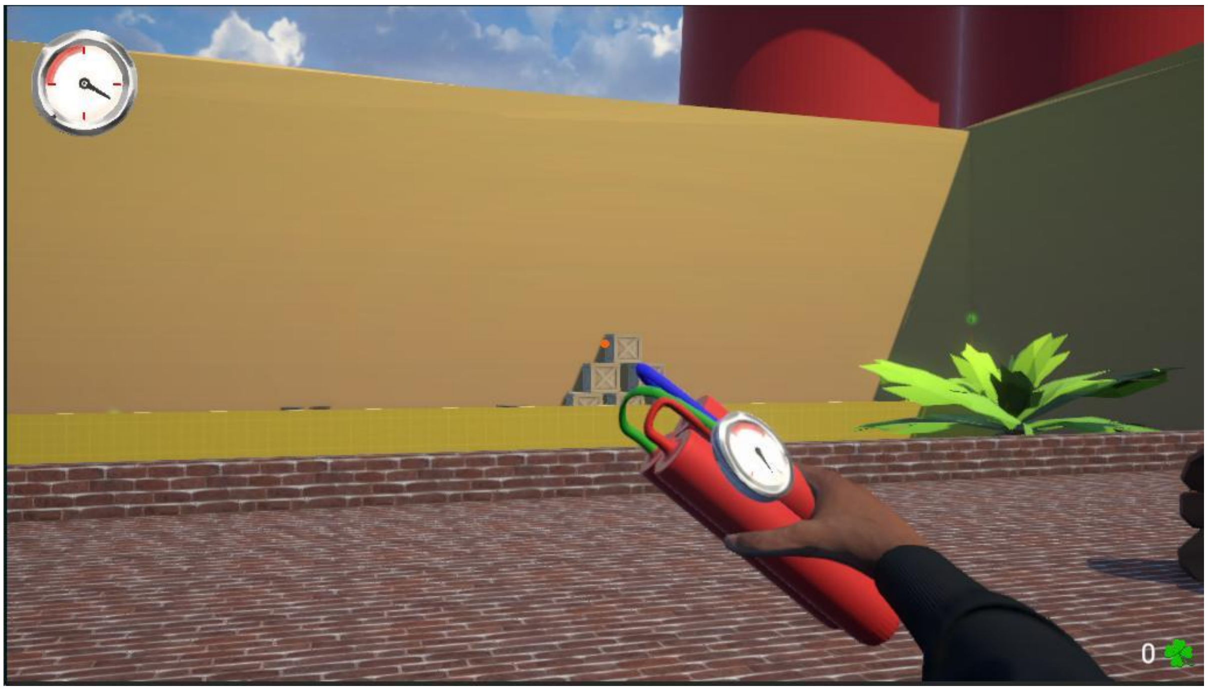
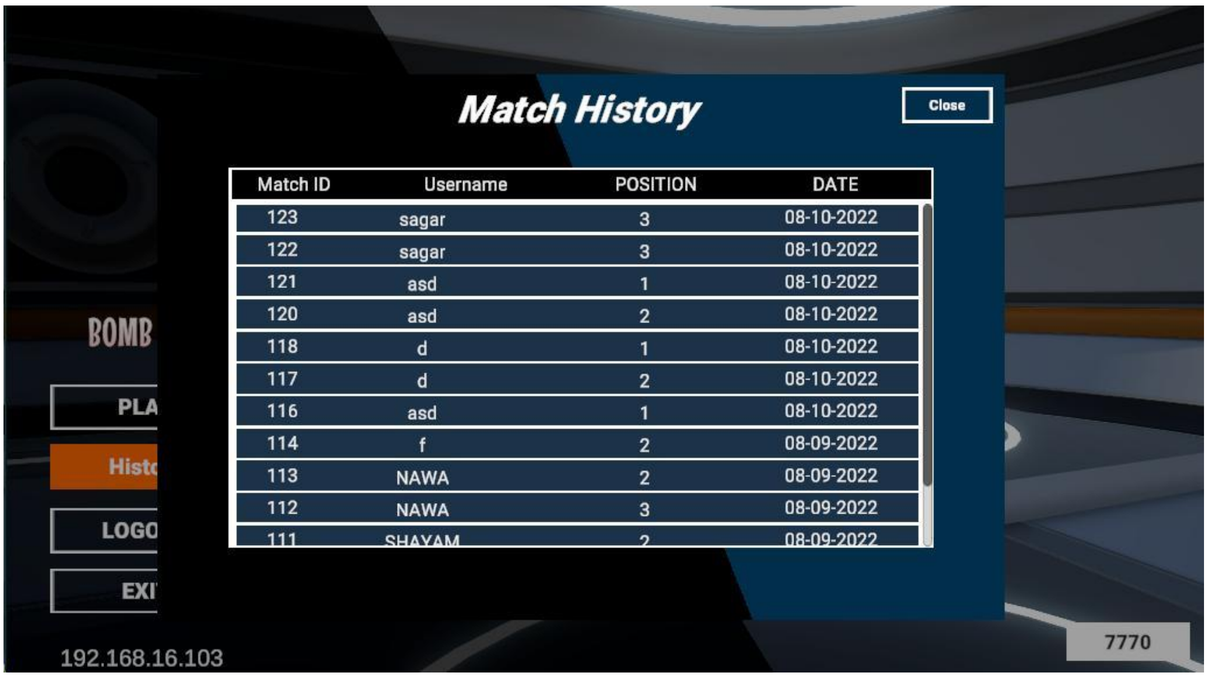


Figure xi: Main game scene



Match History

| Match ID | Username | POSITION | DATE |
|---|---|---|---|
| 123 | sagar | 3 | 08-10-2022 |
| 122 | sagar | 3 | 08-10-2022 |
| 121 | asd | 1 | 08-10-2022 |
| 120 | asd | 2 | 08-10-2022 |
| 118 | d | 1 | 08-10-2022 |
| 117 | d | 2 | 08-10-2022 |
| 116 | asd | 1 | 08-10-2022 |
| 114 | f | 2 | 08-09-2022 |
| 113 | NAWA | 2 | 08-09-2022 |
| 112 | NAWA | 3 | 08-09-2022 |
| 111 | SHAYAM | 2 | 08-09-2022 |

Figure xii: Game history

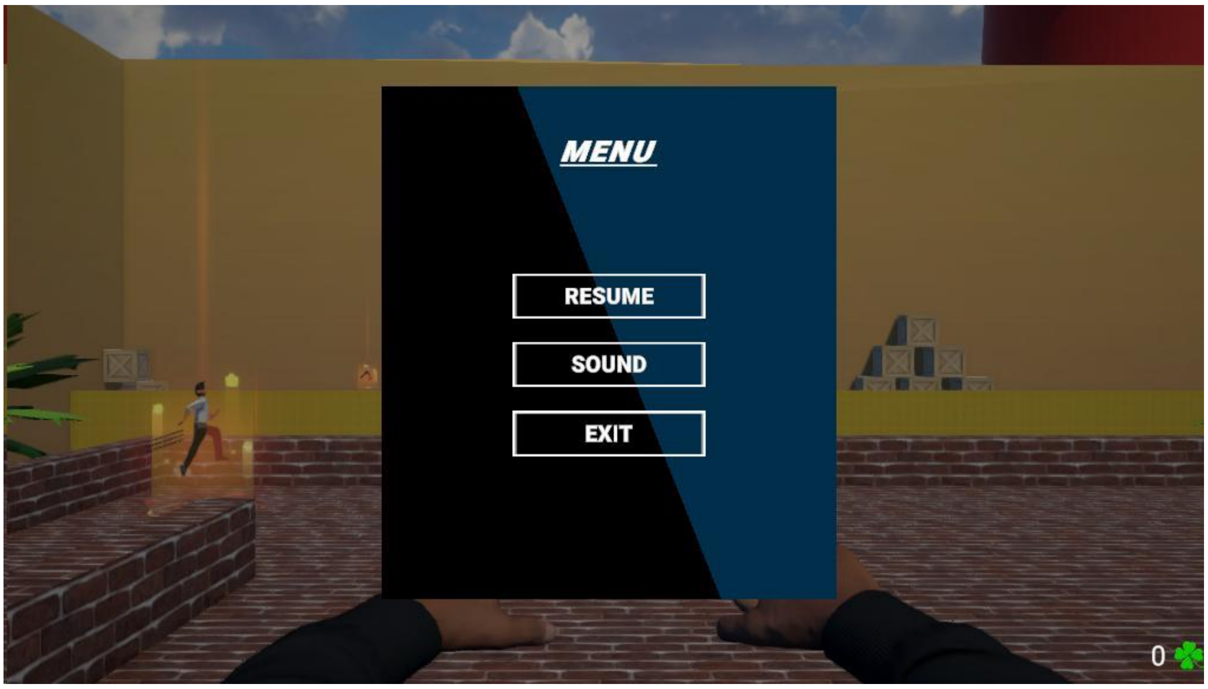


**Figure xiii: Menu**

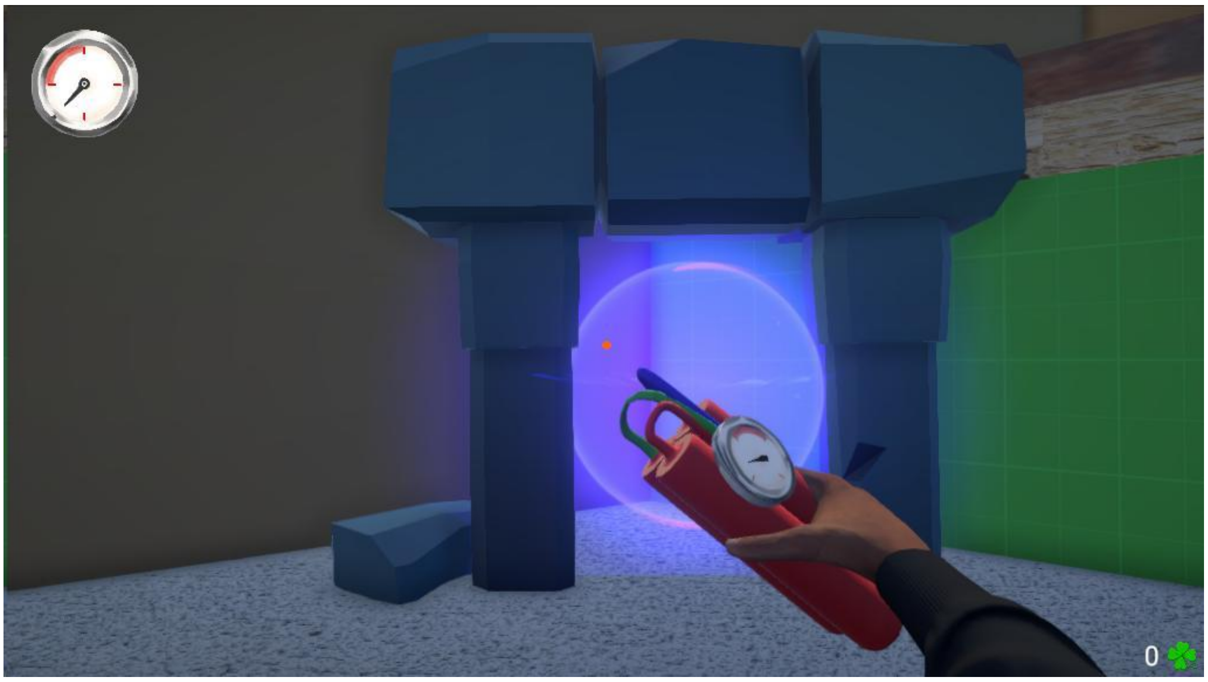


**Figure xiv: Teleporting mechanism in game**

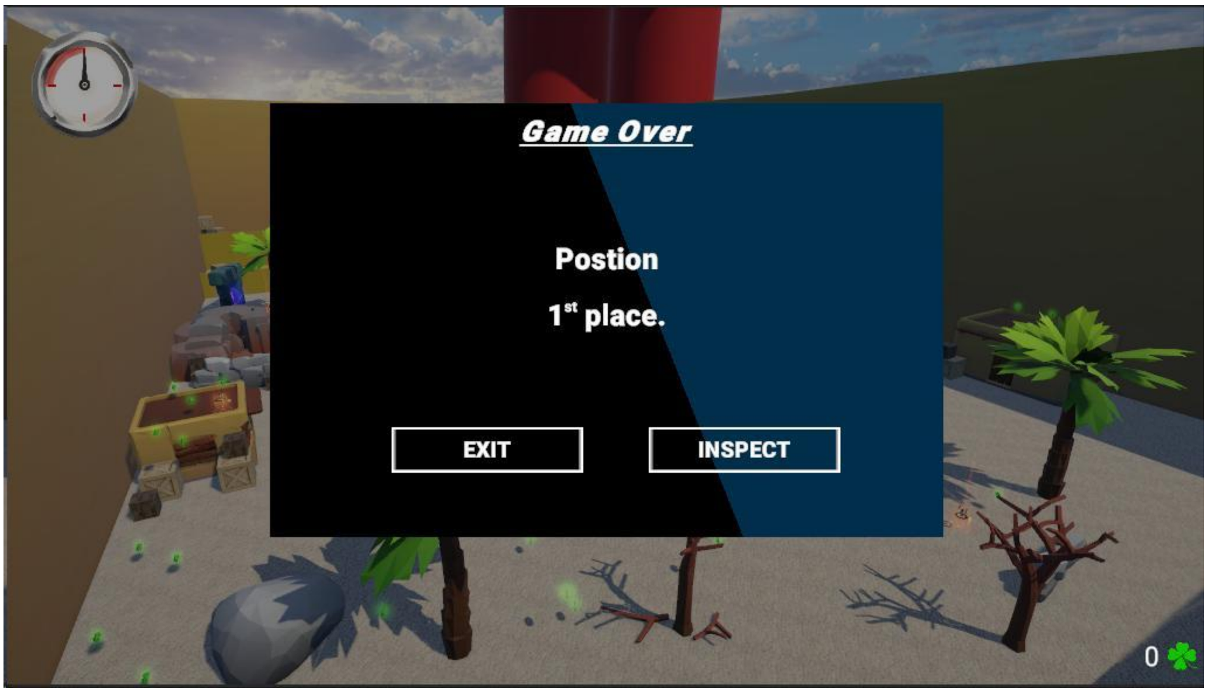


**Figure xv: Game over scene**