Striking Legends

Session 2017-2021

By

Manzoor Turakhel
Hassan Shahzad

Bachelor of Science Software Engineering

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
July, 2021

Striking Legends

Session 2017-2021

By

Manzoor Turakhel
Hassan Shahzad

Supervised by

Sir Shehzad Saifullah

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
July, 2021

Striking Legends

By

Manzoor Turakhel (8949)
Hassan Shahzad (8662)

CERTIFICATE

A THESIS SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIRMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING

We accept this dissertation as conforming to the required standards

(Supervisor) Mr. Shehzad Saifullah	(Internal Examiner)
(External Examiner)	(Head of the Department)
(Coordinator FYP)	(Approved Date)

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
July, 2021

Dedication

We dedicate this project to all our teachers and parents as well as our supervisor who motivated us and helped us in completing our project without their support we would not have done it.

Declaration

We hereby declare that we are the sole authors of this dissertation. The work submitted in this dissertation is the result of our own research, except where otherwise stated.

Manzoor Turakhel
Hassan Shahzad
July, 2021

Acknowledgment

First of all we are really thankful to Allah the most gracious and most merciful. We would like to express our deep felt gratitude to our supervisor Shehzad Saifullah for his patience and guidance. We would also take this opportunity to thank our parents, our friends and our school, college and university teachers, without their guidance and support we would not have reached this stage of our life. We are also very thankful to our brothers and sisters for their continuous love and support through a difficult and trying period of our life. We would also take this opportunity to thank our other teachers for their special support and motivation and for their kind help and guidance. We are grateful to all those people who have helped us a lot with their sincere and continuous support. We want to dedicate this thesis work to our parents who has been a source of inspiration, selfless love and devotion all our life.

Manzoor Turakhel

Hassan Shahzad

Abstract

Video games are a source of entertainment and the gaming industry is one of the fastest growing industry in the world. The quality of games have improved since they have moved from single player offline to multiplayer online games because multiplayer games provides more interaction and competition than single player offline games.
In this report we will explain Striking Legends which is an online multiplayer game. Striking Legends is first-person shooter (FPS) shooting game and is based upon Death Match genre of multiplayer games where players compete against other players of the same level and kills each other in four minutes time. The targeted audience for this game are Esport enthusiasts whom are looking to compete against other players instead of Artificial Intelligence (AI) players. In multi-dimensional maps. Players can switch from one map to another map using portals. By keeping it multi-dimensional, players can strategize against other players and move to other map for entertainment.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

2D	Two-dimensional
3D	Three-dimensional
VR	Virtual Reality
GPS	Global Positioning System
WBS	Work Breakdown Structure
VS	Visual Studio
AI	Artificial Intelligence
Esports	Electronic Sports
FPS	First Person Shooter
HCI	Human Computer Interaction
PUN	Photon Unity Networking

Chapter 1

Introduction

1.1 Overview

Games engage people for amusement and entertainment usually but they can be used for educational purposes as well. These activities can be in physical form such as playing cricket, football and they can be in the form of video games. Video games have evolved over time from simple 2D games to modern 3D and VR games and now it's one of the biggest entertainment industries in the world with more than 2.8 billion gamers around the world and is one of the fastest-growing industry as well [1]. Video games are available for different types of platforms such as the following:

1. PC games.
2. Console games.
3. Mobile games.
4. Browser games.
5. Arcade games.
6. Virtual Reality (VR) games.

1. Computer games are games that are specially designed for personal computers which involve players interacting with them via a monitor/screen. This is one of the most famous gaming platforms for game developers.
2. Console games are games that are played on consoles connected to a television or monitor equipped with a special electronic device that lets the gaming device connect to consoles. Consoles are also one of the most famous gaming platforms for game developers.
3. Mobiles games are games that are specially designed for mobiles because of their unique features such as accelerometers, GPS and other functionalities which give the support for Augmented reality (AR) gameplay. The demand for mobile games

is increasing day by day and now it has become one of the top gaming platforms for game developers.

4. Browser games are games that are designed to be played on web browsers across multiple devices by providing a cross-platform environment.

5. Arcade video games are those games which are played on devices that are specially designed for only one game.

6. Virtual reality games are games designed for VR devices that are mounted to player's heads by providing stereoscopic screens and the ability of motion tracking which submerges the player into the virtual environment by responding to their head's movements.

Many others such as Handheld consoles, emulation, cloud gaming and backward compatibility games.

1.2 Background

PC games have been around since 1960s and since then, thousands of games are developed and are still under development [2]. Games aren't just for fun, there are other objectives in building of games, that's the development of reflexive and problem-solving skills and it is a great way of learning things visually and interactively. Day by day games are moving towards competitive games because competitive games have a longer and unending life cycle compared to story oriented games which are limited to few years after their release date. Some Multiplayer games that are famous [3]:

1. Call of Duty: Warzone.

2. Valorant.

3. Apex Legend.

4. CS:GO.

5. Overwatch.

6. Among Us.

Since Esport is a new and emerging sport in Pakistan, by developing this game, we'll add a local game to Esport list. Besides entertainment, games are also used for economic strengthening. Games and gaming development companies such as Ubisoft, Rockstar

and Tencent make millions and billions of money by selling games and providing in-game purchases. Game development also provides thousands of job opportunities which also has a direct impact on a country's economy. These are the industry roles involved in games.

1. Publishers are the companies that take games from developers and bring it to markets.
2. Distributors are companies/peoples involved in distribution process of games.
3. Retailers include shops, online retailers and others who are involved in selling these games to consumers.
4. Hardware Manufacturers are the companies involved in making the hardware for games.
5. Journalism are the people who publish game-reviews, covering game before their release time to bring excitement before release time.
6. Gamers are the consumers and players of the games who play these games.

1.3 Motivation

Playing games has always been one of our favorite hobby from childhood; therefore we wanted to develop our own game. So, now we have the skills and motivation to learn and then develop our own game. Since Esport community is growing day by day all over the world and it's new in Pakistan so we want to contribute to Esport community of Pakistan and industries across the country by developing a multiplayer game which can contribute to Esport community.

1.4 Scope and Objectives

1. Making a multiplayer game.
2. Providing parallel maps.
3. A game with Deathmatch genre of multiplayer games.
4. Providing player statistics.
5. Our own character model.
6. Skill-based matchmaking.

1.5 Thesis Outline

Chapter 1 covers the introduction, background study, motivation and scope and objectives. Chapter 2 is all about game planning and the matters related to project planning such as project deliverables, WBS and the risks involved in the project and others. Chapter 3 is about game specification and functional and non-functional requirements of the game. Chapter 4 covers game design and all the required tasks for designing a game. Chapter 5 covers the tools used for developing Striking Legend. Chapter 6 is all about implementation, it covers all the details of implementation from interfaces to gameplay. Chapter 7 covers the testing part of Striking Legends including all the possible test cases needed for the project and how the quality of the project was improved by removing bugs and others.

Chapter 2

Game Development Plan

In this chapter, we will discuss all the activities for managing our project such as activities estimation, Work Breakdown Structure (WBS), Resource Allocation, Activities Risks, Cost Estimation for Resources, Gantt Chart, Network diagram used for activities estimation, process model and a sequence diagram.

2.1 Products and Activities

In order to develop Striking Legends we needed a life cycle model best suited for the project so, we chose two different prototyping models each suited for different purposes in Striking Legends. The first one is Evolutionary Prototyping which serves as a base and the second one which is Throw Away Prototyping that is used for uncertain tasks and testing different strategies for the game. Evolutionary Prototyping is similar to Iterative model. In Evolutionary Prototyping we have many builds and each build corresponds to a certain time in our development phase because games evolve over time, they aren't made perfectly at once. Every build has its requirements analysis phase and quick design phase based on the first two phases we build a prototype and the prototype is tested and evaluated by users if any fault is found out we refine the prototype based on user evaluation and repeat this process until the final product is implemented. We are going to have 4 builds. In order to put this model we will describe a scenario for it. In the beginning we were unsure of all the requirements so, we gather requirements for Striking Legends then we design 3D models and scenes for the requirements after designing we build a working prototype according the requirements and quick design such as a fighting scene after the prototype is built we test the prototype using user evaluation as a testing technique if something is wrong we refine the prototype and then add other features and functionalities until the game is completely developed.

2.2 Activities Estimation

Project estimation is done using estimating the time and efforts needed for a project. Both of these are vital in order it to be successful. Most project failure is due to inaccurate estimation of time and efforts because miscalculating the time can make a project delay and miscalculating the efforts makes it out of budget which leads to failure so, to save a project from failing is to make accurate estimates based on different methods provided in project management one of them is Network diagram. Figure 2.1 shows the network diagram and table 2.1 describes the activities and their precedents.

Table 2.1: Activities Estimation

Character	Activities	Duration	Precedents
A	Requirement gathering	6 weeks	
B	Character and Weapon Modelling	10 weeks	A
C	Scene Creation and animations	9 weeks	A
D	making user interface/Main Menu	3 Weeks	A
E	Connectivity to network	5 weeks	D
F	Synchronization of everything	4 weeks	D, E, C
G	Player management system	5 weeks	F

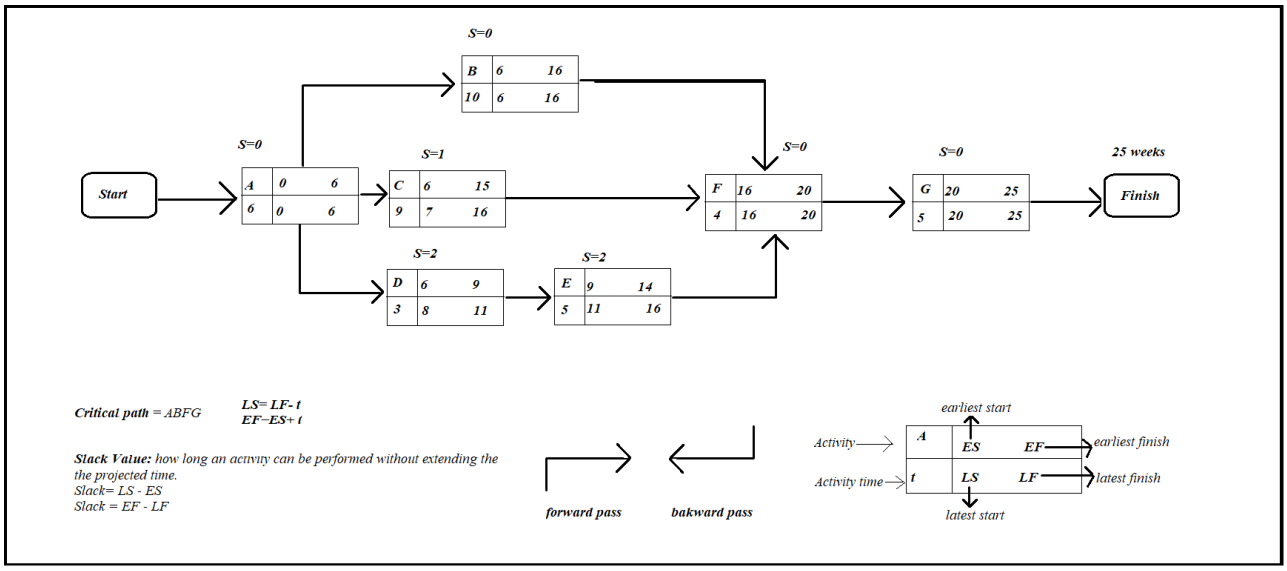

Figure 2.1: Network Diagram

The first activity is requirement gathering because requirement gathering does not has any precedents. The earliest start (ES) is week 0 because it's the first activity and the earliest finish (EF) is week 6. The EF is derived from the formula A and from backward pass the first activity can not be completed in longer than 6 weeks and the ES is week 0 and it's derived from the formula B.

$$Formula\ A = EF = ES + t$$

$$Formula\ B = LF = ES - t$$

Activity B, C, D can start parallel as each of them have only one precedent which is activity A because activity B is character modeling and C is Scene Creation and D is menu and interface. Each of them can start separately because character modeling is done in Blender, Zbrush, Maya or any other modeling tool. Character modeling involves Box modeling, Sculpting, Retopology, UV Unwrapping, Baking and at last texturing and animation; therefore it takes more time than other activities. The ES is week 6 and EF is week 16 and from backward pass the latest finish (LF) is week 16 and the latest start (LS) is week 6 and the slack value is 0 which is defined as formula C. The slack value defines how long an activity can be performed without extending the project time so an activity can't be delayed without the time allocated to it else it will result in delaying the project.

$$Formula\ C = S = LS - ES\ or\ S = EF - LF$$

Activity C is scene creation and the scripting for different scenes such as parallel arenas in the case of Striking Legends and applying the movement and health and other. Scenes and scripting for these scenes are the 2nd most time-consuming activities. The Earliest it can start is week 6 and EF is week 15. Activity S is menu and UI interface which includes connection to the server and the creation of rooms. Rooms differs from custom to random and joining the rooms based on skills and overall record of each player.
Activity E involves network connectivity because Striking Legends is mainly based on multiplayer so it needs internet connectivity and loading the data onto the network needs work so the earliest time it can start is week 9 and EF is week 14 and the LF is week 16 and LS is week 11.
Activity F is synchronizing the scene and player movement in the game when scenes and players are loaded onto the cloud server it needs synchronization such as the movement of players updating the kills and health across the network to all the players. The earliest time it can start is week 16 and the earliest finish is week 20 and from backward pass the LF is week 20 and the LS is week 20. When all of these activities are done overall

management system for players will be implemented where each player has points, email and password stored in PlayFab. This is done at the end because our main goal is on network. Management system is an extra feature so the ES time is week 20 and EF time is week 25 and from backward pass the LF time is week 25 and LS time is week 20.

Critical path: The activities with zero slack values are called critical activities and these activities form a critical path. So those activities with zero slack values are ABFG. Activities with S=0 are the activities which can't be delayed without extending the project time and delaying the project time and the activities with the slack value above 0 are activities that be delayed up-to their slack value without delaying the project time.

2.2.1 Work Breakdown Structure (WBS)

Also known as PBS product breakdown structure which is one of the main parts of project management which distributes the project into smaller parts so that it's easy to understand by team members. WBS divides activities into sub-activities which can be more manageable by management team and development team. Figure 2.2 shows WBS of Striking Legends.

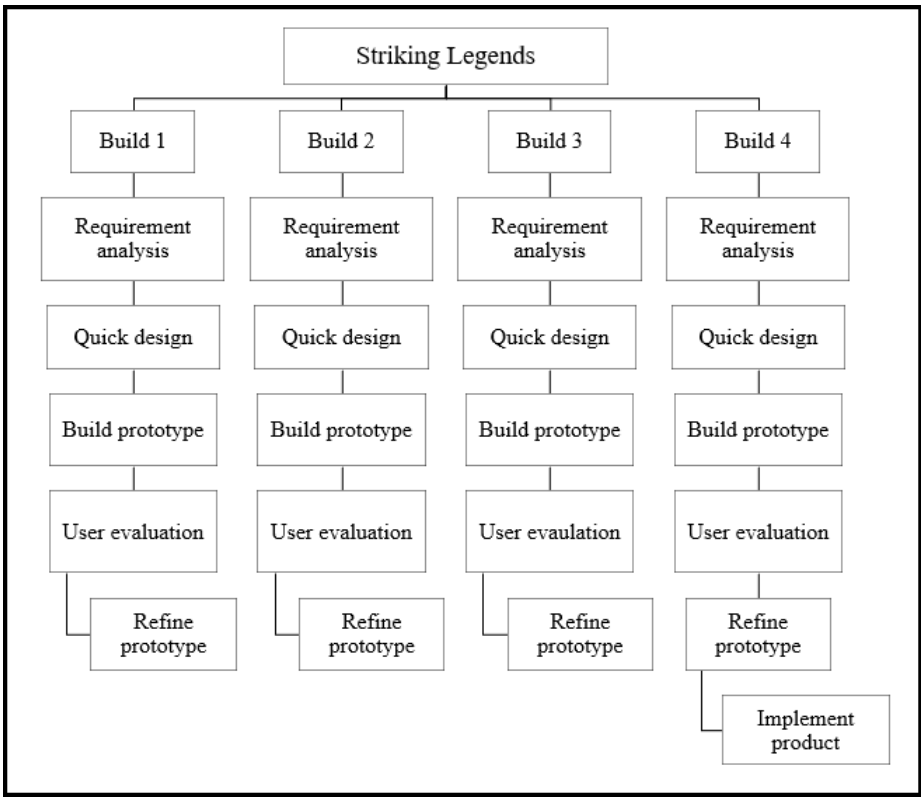

Figure 2.2: Work Breakdown Structure (WBS)

2.3 Activities Risk

Table 2.2 describes risks and the strategies for risk reduction along with the contingency.

Table 2.2: Activities Risk

Risks	**Risk Reduction and Contingency**
Group member sickness	**Risk reduction:** Complete activities before deadline to have some extra time in case of emergency. **Contingency:** Work overtime.
Corrupted hard drives	**Risk reduction:** As this has happened with us every group member needs to have backup copy online in Google drive or using a portable hard drive. **Contingency:** Recovering the corrupted files.
Resource shortage	**Risk reduction:** Keeping an additional laptop and opting premium softwares. **Contingency:** If needed using pirated software because we don't have the money to pay for them.
Stuck in implementation	**Risk reduction:** Searching it online and joining gaming discord groups for such implementation problems.As this was the case which happened with us when we couldn't apply the animation we made for the character to it. **Contingency:** As some implementations are independent leave it until group member can solve it.
Unclear Requirements	**Risk reduction:** Clearing the requirements before implementation because after implementation a lot of time would have passed. **Contingency:** Arranging a meeting between group member and clearing the requirements before moving forward.
Deadline	**Risk reduction:** Working ahead of the schedule. **Contingency:** Using already available assets and material on the internet.

2.4 Gantt Chart

Our model contains four iterations. As shown in the figure 2.3

- The first build will serve as a skeleton which might have basic functionality of game.
- In the second iteration of the build we will add more functionality to our game and also synchronize these functionalities with the previous one.

- In the third iteration of the build we will add functionality with icons, texture and etc.
- Last phase we will mostly focus on polishing the game, filtering the game, testing and completing the missing requirements.

		Octuber and November				December and January				February and March				April and May			
S.no	Phase	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
1	Requirement Analysis																
2	Quick Design																
3	Build Prototype																
4	User Evaluation																
5	Refining Prototype																
6	Implement Product																

Figure 2.3: Gantt Chart

2.5 Process Model

We have chosen evolutionary prototyping because our game will evolve over time and since evolutionary prototyping helps in testing the product in every evolution of the build by showing the result and improving it in the next phases. Evolutionary prototyping is about how to create a model which is refined over time. Like stages of evolution, each stage of the process involves resolving any issues and implementing any changes to the prototype. In evolutionary prototyping, the first build serves as a heart or base for the game which will evolve over time by making changes and doing improvements into the base. Figure 2.4 shows the process model used for Striking Legends.

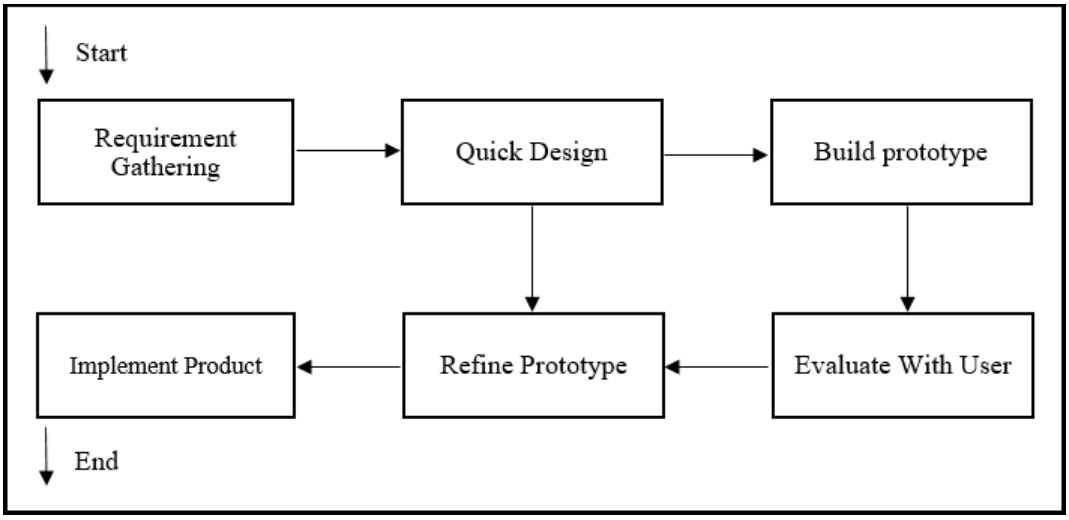

Figure 2.4: Process Model

2.6 Sequence Diagram

Sequence diagram also known as event diagram or event scenario diagram which describes different scenes or events that occur simultaneously and how messages are exchanged between these objects. Figure 2.5 shows the sequence diagram of Striking Legends. The figure shows how Striking Legends works and how the player interacts with different scenes of the game.

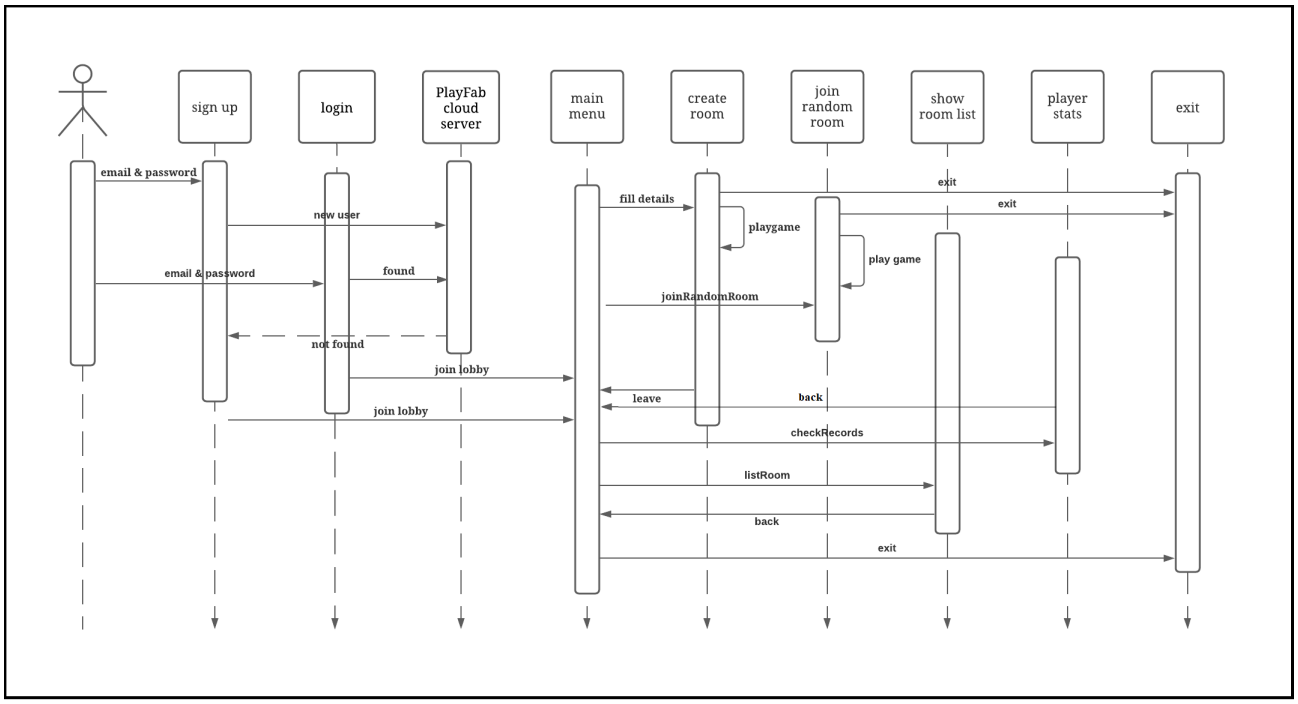

Figure 2.5: Sequence Diagram

Chapter 3

Game Specification

These requirements are noted in SRS which stands for software requirement specification. SRS contains all the requirements for a specific software in case of Striking Legends it will have functional and non-functional requirements.

3.1 Functional Requirements

Requirement specifies functions of a system what the system does and what the system is supposed to do they can be in form of input and output so, in general functional requirements helps in identifying the behavior of the software. The followings are the functional requirements for Striking Legends:

- Login Module for logging registered users.
- Signup Module for registering new users.
- Create room option.
- Join random room option.
- Player management system used for managing the player statistics.
- Health regeneration system for regenerating player's health within few seconds when out of fight range.
- Respawning system for respawning the player in a room multiple times in multiple places.
- Level based matchmaking using player's points.
- Damage system uses different damage points for different weapons.
- Exit system used for exiting from the game during game or during join room weapons and different damage points on different body points.
- Room lists Showing the number of rooms with a back option.
- Return to lobby during game and in other scenes.

3.2 Non-Functional Requirements

Non-functional requirements of a project are related to its quality attributes such as performance, security, usability and etc. The following are some of the non-functional requirements of Striking Legends:

- Usability: The game is easy to use because we have made simple interfaces and followed the standard of shooting games for character controller.
- Performance: The game performs very well even it's on network because we have used different methods to make it lightweight which boosts the performance.
- Security: The game is secured because we are using cloud server as a database.
- Reliability: The game is reliable because it is bug free.
- Availability: The game has 24/7 available because its online and multiplayer.
- Recoverability: The game recovers in case of network issues or pings problem.
- Maintainability: The game is maintained in case of bugs or errors which makes it future proof.
- Interactivity: Keeping the player engaged.

3.3 Use-Cases

Use case diagram demonstrates how actors interact with the system. In other words, it shows the possible interaction between the system and external entities. Figure 3.1 Shows the use-case diagram of Striking Legends.

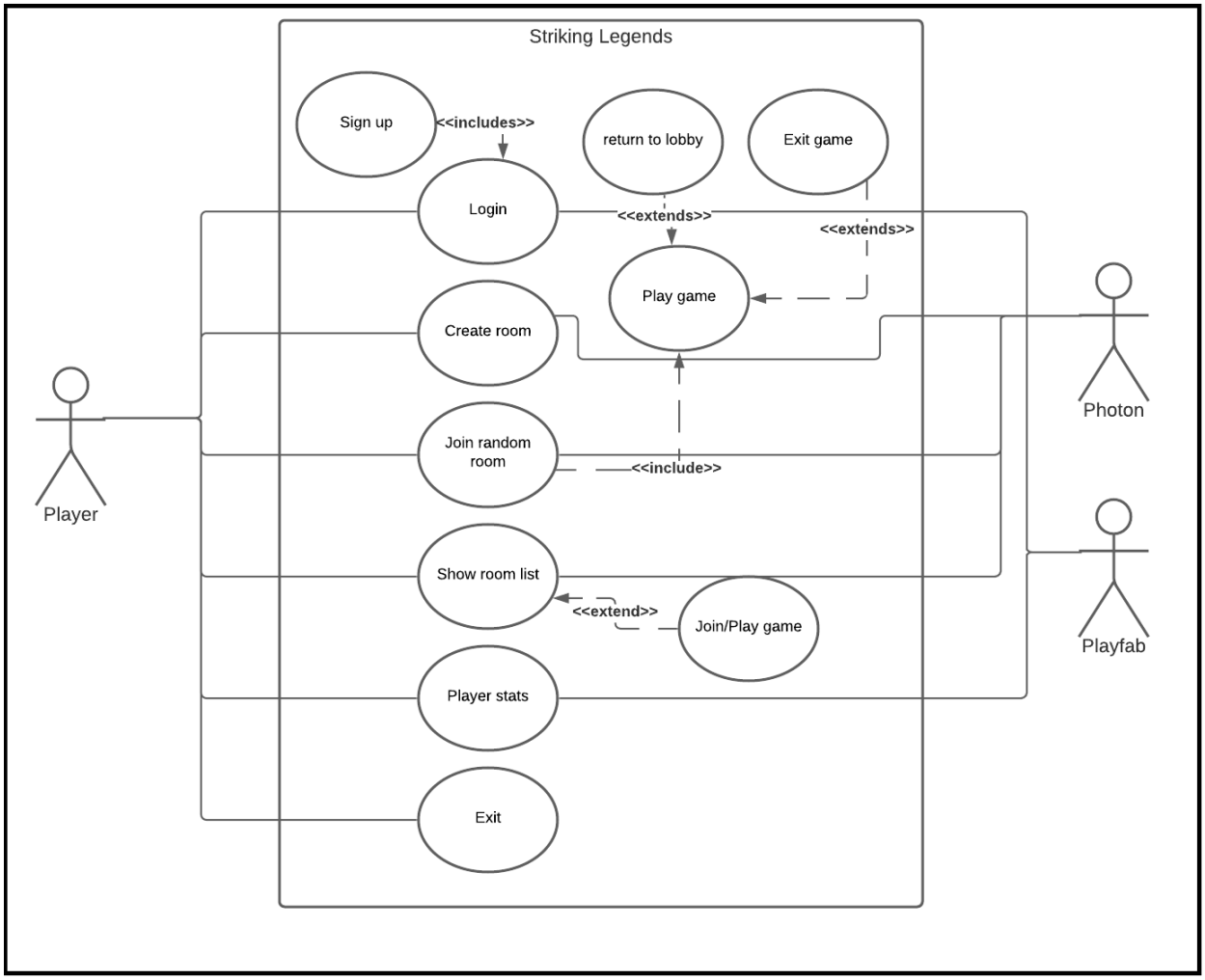

Figure 3.1: Use Case

Chapter 4

Game Design

Game design is the art of designing and applying aesthetics for creating games by bringing imaginations to life video games. The focus is on bringing interaction between different game objects. Imaginations are the stories, Characters, Environments and different games objects. Robert Zubek a game designer defines a game such as this [4].

- Mechanics and systems, which are the rules and objects in the game.
- Gameplay, which is the interaction between the player and the mechanics and systems.
- Player experience, which is how users feel when they're playing the game

Based on games design, Games are divided into multiple genres. Every game developed falls into one of these genres:

- Action games.
- Adventure games.
- Role-playing games.
- Simulation games.
- Strategy games.
- Sports games.
- Massive Multiplayer Online (MMO)

Action games: They are games where players participate in physical challenges such as hand-eye coordination and reflexes. It includes many sub-genres such as shooting, fighting and others.
Adventure games: These games engage the main player in stories and puzzles. Adventure games genre is inspired from media and film documentaries. In Adventure game the main player plays the role of a protagonist where the player explores the story and solve puzzles.

Role-playing games (RPGs): They are games that involve the player into playing a fictional character such as a warrior, prince, king and others where the character is guided through a narrative or structured decision making system for character development.
Simulation games: They are games designed to emulate real or fictional reality. real life reality can be in form of driving, training, management and others.
Sports games: They are games which are based on real-sport such as cricket games, football games and others.
Massive Multiplayer Online (MMO) games: These games are specially designed to be played over internet and network and has multiple sub-genres. MMOs have multiple modes to play from solo to teams where players compete against each other. Some of these MMOs are as follows:

- Get the Flag is a type of multiplayer game which is based upon ancient wars strategy where player needs to take other teams' flag in order to win the match.
- Destroy the bomb is a type of multiplayer game where player needs to find the bomb and dispose the bomb before it explodes.
- Hold position is a type of multiplayer game where players defend their position until the objective is achieved.
- Deathmatch is a type of multiplayer game where players fight each other to get maximum kills in a specific time. Our game is also based on deathmatch genre.
- Battle Royale is a type of multiplayer game where player starts with minimal equipments then the player starts collecting loot and then the player try to eliminate other players while staying inside a shrinking area called "Safe Zone" until the player is the last survivor.

4.1 Game Concept Overview

Striking Legends is a first-person shooting game based on deathmatch genre of multiplayer games. The game take place in parallel dimension arenas connected to each other via a portal where players can fight each other in four minutes allocated time. It's a solo-based multiplayer game where different players in the same room fight each other in 4 minutes time where they can be re-spawned multiple times in multiple places defined in spawn points.

4.2 Game Mechanics

Our game has the following mechanics.

- Striking Legends is FPS multiplayer based on deathmatch genre.
- Players will fight each other in order to win in allotted time.
- Players can choose from multiple rooms to join.
- Player can randomly join room based on level.
- Players can move between 2 arenas using portal.
- Player can regenerate health if they can keep themselves save for few seconds.
- Different weapons have different damage points.
- Players have management system where their points are saved.

4.3 Game Characteristics

1. **Resource allocation:** The game doesn't require a lot of resources such as ram and hard drive space because we have optimized the game for this by making the 3D modules low poly and applying low quality textures to make it optimized.
2. **Smooth Performance:** Striking Legends performs smoothly. Smooth performance is achieved by synchronizing the actions such as animations and gun firing and other actions and making the character and assets low poly.
3. **Graphics:** Striking Legends has good graphic. Good graphics are achieved by baking the models on high poly models and doing re-topology of models to make them low poly and make them look as good as high poly models.

Chapter 5

Tools

Tools are important for any software project especially games because making everything require proper tools. One can't make a simple software without the required tools. So, tools are the things which help in the process of developing and making something easier. Therefore we needed proper tools to choose, In order to develop Striking Legends effectively and efficiently. So, to choose these tools we decided to only pick the tools which can well fit according to our needs and our project; therefore we decided to choose the following described tools.

5.1 Unity

Unity is a cross-platform game engine which was developed by Unity Technologies in June 2005 at Apple Inc [5]. Unity is developed using C++ and uses C-sharp as a scripting language. Initially Unity was only released for Mac OS but with the passage of time Unity extended it support for other platforms such as Desktop, Consoles, Mobile and VR. Popular mobile games such as Call of Duty: Mobile and Pokémon Go were developed using Unity engine. Unity can be used to create 2D, 3D, simulations and other types of games. Even it's used in the non-gaming industry such as films with the introduction of Cinemachine tool in 2010 [6]. We have also used Unity because of its networking packages and its large online community and the tutorials available for this tool in Youtube and on other platforms. The other reason for choosing Unity was that it's not graphic intensive compared to Unreal engine because our main focus is on networking. Unity has multiple versions from 1.0.0 which is the first version to 2021.1.14 which is the latest version but we have used 2020.1.9f because it was the latest and stable version at the time we started the project and we are still using the same version because upgrading to newer versions needs changes to the project which we couldn't afford therefore we sticked to that same version and we are using Unity Personal version because other versions require payment.

5.2 Blender

Blender is an open-source and free 3D computer graphics software developed by NeoGeo in January 1995 using C as an in-house application [7]. Which later on in 1998 was released publicly as a freeware software [8]. Blender is used for creating animated films, visual effects, 3D printed models, computer games, VR, simulations, motion graphics and even video editing and many more. We have used Blender for 3D modeling and animations because of the features it provides and also it is a free software instead of using Maya or any other 3D graphics tools which are paid. The other reason for choosing blender instead of other 3D graphics software is lightweightness which makes it less resource hungry. There are many versions of Blender but we have used 2.91 version.

5.3 Visual Studio

Visual Studio is an IDE developed by Microsoft in 1997. Used for the development of Desktop Applications, Websites, Mobile Apps and others. It supports 36 different programming languages and its integrated debugger works as a source-level debugger and machine-level debugger and has many other features such as code profiler which is used for GUI applications and many other features. We have used Visual Studio Community 2019 because it's the free version of Visual Studio. We have chosen Visual Studio over other IDEs because it provides IntelliSense and Code Refactoring.

5.4 Adobe Photoshop

Adobe Photoshop is a raster graphics editor developed by Adobe Inc and released in February 1990 [9]. We have used Photoshop for texture editing, making of background images for Striking Legends, icons and Logo design. As Adobe Photoshop has many versions, but we have used Adobe Photoshop 2020.

5.5 C# (C Sharp)

C# is a general purpose and multi-paradigm programming language developed by Microsoft in 2000 [10]. We have used C# as a programming language because Unity only supports scripting in C#.

5.6 Photon Server

Photon server is a real-time cloud server used for hosting multiplayer games. Photon provides multiple services such as PUN, BOLT and QUANTUM [11]. PUN is also integrated with Unity which makes the multiplayer system easier compare to other Unity networking packages. We have used PUN2 for making Striking Legends a Multiplayer game because it's easier compared to other networking packages and because it provides up to 20 player free service called 20CCU [12].

5.7 PlayFab

PlayFab is a multiplayer service which provides complete backend platform for live game. Founded by Microsoft in 2011 [13]. We have used PlayFab for our player management system where Players accounts and their statistics are saved.

5.8 Lucid Charts

Lucidchart is a web-based app which provides different drawing facilities such as flowcharts, UML diagrams and others. We have used Lucidchart for making our Sequence and Use-case diagrams.

Chapter 6

Implementation

This chapter will cover all the implementation details from the start to a full multiplayer game along with the process model and the issues we have faced during the development phase. Implementation includes the tools used for the game and the deliverable of these tools and the step taken for developing these deliverables. Deliverables varieties for different tools. For instance, the deliverables of Blender are animations, models, textures and the deliverables of Unity are builds. The following sections describe details of implementation.

6.1 Character Building Process

The character used in games are designed using 3D modelling software such as Maya 3D, Blender and many more. we have used Blender for character modelling because it's free. Character modelling requires a lot of hard work the following steps are done in order to make a 3D character for a game from a single cube to a full 3D character:

1. Blocking out: Blocking out is the first step for making a gaming character where base shape is made using simple cubes, spheres, cylinders and other basic elements.

2. Sculpting: It's is the process of sculpting a model where polygons are manipulated to make a proper and detailed shape. Sculpting requires a lot of skills and time.

3. Re-topology: It's is the process of making the existing model with a better and enhanced topology where number of polygons are decreased from million to thousands. It's the process of converting high poly character to a low poly character which can be used in games.

4. UV-unwrapping: It's the process of wrapping a 2D image to 3D objects. This is done u in order to apply texture to a 3D object/models by defining coordinates on the 3D objects and marking them seam and then unwrapping them for applying textures.

5. Baking: It's the process of transferring details from high poly character to a low poly character in order it the low poly character look as good as the high poly character.

6. Animations: Applying animations to a 3D character needs a rig. A rig is bone structure used for a 3D model. Animations is possible only through rigging where these bones structure are used to control the 3D model.

6.2 Animation Export Process

Animations are made in 3D modelling software in our case it was Blender. To make the animations work on our models we had to export the model along with the animations into an FBX file although unity natively imports blend file but the FBX was an easy way to do it. Striking Legend has 14 animations used for different purposes from simple walking animation to weapons reloading animations and more.

6.3 Health System

Health is an attribute that determines how much damage can a player take before dying. In many games, they are in form of hit points (HP) represented by health bar, numerical values, icons, heartbeat and others. Striking Legends uses numerical values for health system. A player has 100HP when his health is full and zero when the player dies. Health can also be attached to non-living objects such as cars and other destructible objects which explodes when they reach their maximum HP. Players Health depletes when they take damage and eventually die when their health point decreases to Zero. Every game has its unique health system before a player dies such as red alerts and fading the player consciousness when a player is near death and others but Striking Legends doesn't have such a system because Striking Legends is multiplayer and it's based on the deathmatch genre of multiplayer games. Regeneration of health is also a part of health system because when a player loses some health, he must be able to regenerate back to full health. Every game has a different approach towards regeneration of health, some games use potions and medicine as this was the case in The Witcher series and first-aid kits, pain killer used in many multiplayer battle royal games. Striking Legends uses another approach for health regeneration which is when a player is out of attack range for 5 seconds his health automatically regenerates and a special ability to regenerate health two times faster when the player uses the portal. This regeneration ability to regenerate two times faster is only for 10 seconds after the player uses the portal.

6.4 Damage System

Damage system is one of the important systems in games because damage can determine how a player takes damage and how the player dies after taking damage and even it can be said for other game objects that are destructible. There are different types of systems for a player or any other game object to take damage. Such as using a Melee weapon to damage a player which works such that when the Melee collides with the player or any game object it will take damage. For players it will result in HP loss and for other game objects it can be in form of blasts, smoke, fire and others.
Damage can also be in the form of falling from a high point called fall damage, being in the radius defined for a Grenade explosion, getting drowned in water and others. Striking Legends uses bullets for damaging other players because it's the simplest form of giving damage and it's a suitable scenario for Striking Legends because we don't have the environment and game design for other kinds of damage systems. Taking damage from bullets are not same for different weapons. It varies from pistol to rifles because in reality, their damage is different therefore, we have used different points for different weapons. All rifles have high damage points than Pistols with a drawback of reloading time which is more than pistols. Rifles also vary in damage points such as 1R-50 has more damage points than MG-7. When a player losses 100 HP the player will die and will be respawned to a random position defined in next section spawn points. The player can be respawned multiple times as long as the room is not finished.

6.5 Spawn Points

They are the locations on map where players are spawned when they start the game. Striking Legends has eight spawn points and these spawn points are randomly assigned to each player by the system which helps camping harder for other players. Players are respawned when they die to that specific spawn points randomly. Spawn points are relatively close to each other for the purpose of finding other players easier. Capsule-shaped objects are the spawn points in figure 6.1.

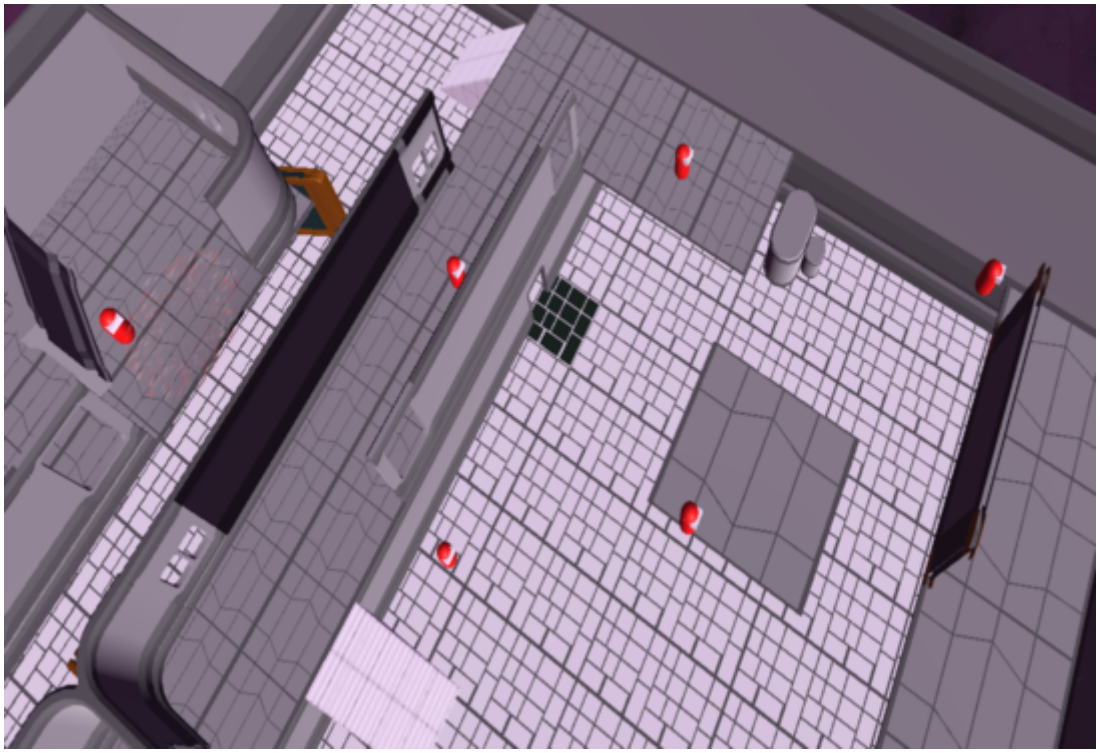

Figure 6.1: Spawn Points

6.6 Portal

Portal means a doorway, gateway or entrance point. So, a portal is used as a gateway for different purposes in different games. Some games uses a portal to unlock different abilities, change players levels, others use it to change locations, dimension such is the case in Apex Legends which uses portal for changing dimensions; Therefore we decided to have something similar which is giving the player a teleportation ability using portal. The portal gives players the ability to teleport from one map to another for fun when they get bored and also a strategy point for players to move from one arena to another. Striking Legends which has parallel dimensions allow players to move from one arena to another arena using portal. When a user uses portal, he/she gets a special ability for 10 seconds to regenerate health two times faster than normal. The portal opens after 15 seconds of spawning time and closes for 10 seconds after 10 seconds the portal re-opens and players can go through portals to switch arenas. Figure 6.2 shows the portal

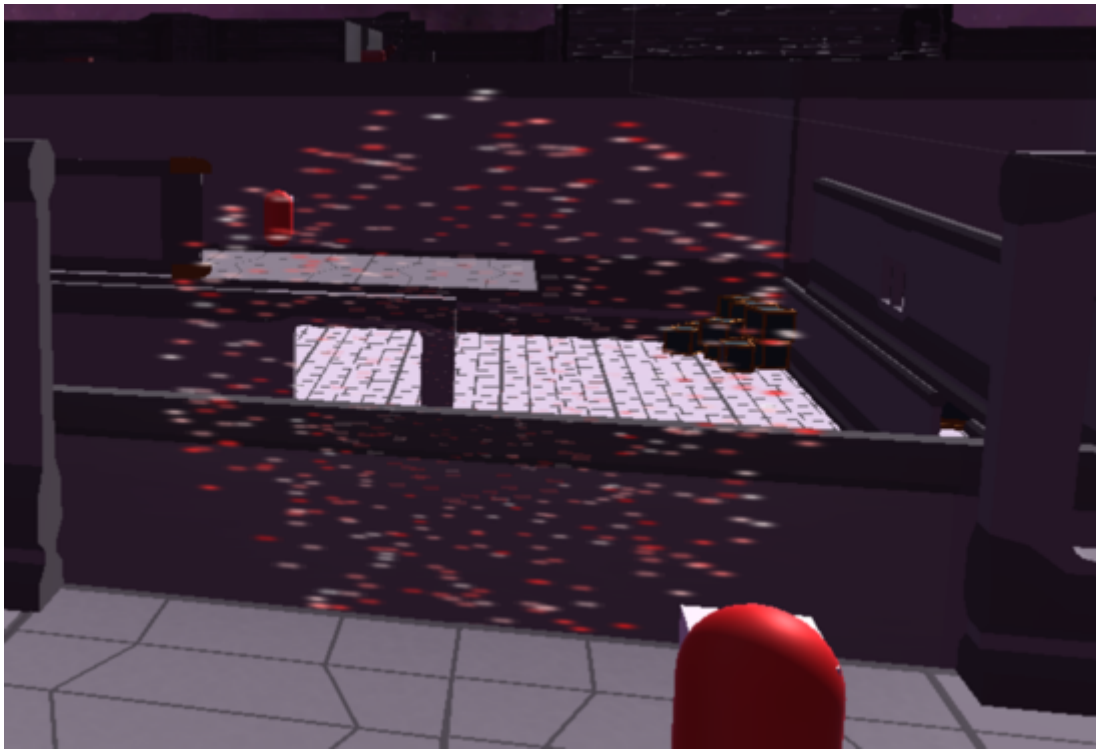

Figure 6.2: Portal

6.7 Skill-based Matchmaking

Matchmaking is the process of connecting different players into a joined Session/game. Most of multiplayer online games has skill-based matchmaking system where players Can play together based on ranks, levels, points, experience (XP). Striking Legend also has a skill-based matchmaking system which is based upon points. When a player signs up for a Game Default points of 1500 are given to the player. Points increases based on kills. When a Player Kills another player, his points are increased by 20. The skill-based matchmaking works Based On an algorithm which looks for the master client points when creating a room and when Another Player searches for a room this player's points are compared to the room masters points $\pm$ 300 when their points match, they are joined together to play if it's does not matches this player create his own room based on his own points where other players can join.

6.8 Game Play Programming

6.8.1 Camera Controller

Camera controller is the system through which the camera in the Game is controlled. Camera controlling system allows the player to look into different direction Using mouse as we see it using eyes by moving it to different directions. Player can see to left And right fully except his back and front face because we have first person camera controller System which does not allow looking into these directions and the camera moves along with the Targeted player to where ever the target goes and allows him to see other players. Camera Controller follows the player along his speed because the normal speed of a player is slow when The player runs the camera automatically moves faster along with the players movements. We have used First Person Controller system for Striking Legends.

6.8.1.1 First Person Controller System (FPS)

To give it a realistic look we made Striking Legends as a first person shooter game because it's how we human perceive the world via eyes. First person is similar to how we see things. Giving it FPS Controller system, the player can't see himself entirely but can see other players' character.

6.8.2 Player Controller

Player controller system is the controlling system used for controlling players movements. Here We will also discuss the controlling devices needed for Striking Legends and how each device is used.

- Mouse Controls
- Key Board Controls

6.8.2.1 Mouse Controls

Mouse is needed for looking into different angels, aiming, weapon changing and firing. Table 6.1 shows the mouse controls.

Table 6.1: Mouse Controls

Button	**Functionality**
Left Click	Firing
Right Click	Aim
Scroll Wheel	Weapon Changing

6.8.2.2 Keyboard Controls

List of different keys used for different tasks. Table 6.2 shows the keyboard controls. All

Table 6.2: Key Board Controls

Keys	**Functionality**
W	Moving Forward
S	Moving Backward
D	Moving right
A	Moving left
C	Crouching
Space	Jump
Left shif + WSAD	Faster Movement
R	Reloading
Numeric keys 1-5	Weapons Switching

of the buttons used for Game controls are according to gaming standards and it follows the rules of Human Computer Interaction. Using these controls player can easily control characters movements because all the control buttons are easier to reach because they are nearer to each other which gives the player full control and confident while playing the game.

6.8.3 Track Boundary Checking

Track boundary checking are the block which does not allow the player to go out of playing area. We have used walls that ensure the player to be in the arena. By colliding with the walls the player is prevented from leaving arena.

6.8.4 Game Physics

Game physics is the process of deploying physics into games by using laws of physics. This is done for the purpose of making games realistic to players. Game physics also called Simulation physics is a close approximation to real life physics but some time game physics changes from real life physics laws for entertainment purposes such as double jumping when there is nothing to jump from, making cars fly, making things invisible and others. Game physics is used for collision detection. Collision detection is used when two physical objects cross each others path such as two players, two cars and many more. There are basically two types of physics simulation.

1. Rigid Body simulation : In rigid body simulation objects are grouped and divided into categories based on their interaction and are not performance intensive.

2. Soft Body Simulation : Soft body physics deals with individual objects in such a way that these objects should behave and interact realistically.

6.8.4.1 Particle System in Game Physics

The explosion is a typical feature of computer games that depict conflict. The simple expedient of repeating the same explosion in each condition was employed in early computer games. In the actual world, however, an explosion can vary based on the terrain, the height at which it occurs, and the type of solid bodies that are affected. The impacts of the explosion can be portrayed as split and shattered components propelled by the expanding gas, depending on the computing power available. A particle system simulation is used to model this. Other physical phenomena, such as smoke, moving water, precipitation, and so on, can be simulated using a particle system model. Individual particles in the system are modelled using the other aspects of the physics simulation rules, with the caveat that the number of particles that can be simulated is limited by the hardware's computational power. As a result, rather than a vast number of fine particles, explosions may need to be treated as a small group of massive particles.

6.8.4.2 Projectiles in Game Physics

Projectiles in Game physics Are objects/elements that travels with high speed such as arrows and gun bullets. This creates a problem because of its high speed sometimes collision is not detected and it goes through an object. Previously this problem was solved using ray cast but now this problem is solved using physical projectiles which can be affected by force and gravity which makes it more realistic because this method continuously detect collision.

6.8.5 Development Process

As Striking Legends is developed using Evolutionary Prototyping functionality are added in different builds because it's the nature of games to evolve over time.

Striking Legends has 4 builds every build has additional functionalities compared to their previous builds. Because it's easier to add functionalities to previous builds which are already tested rather than developing it at once because it's impossible in games to be developed in one build or at once. Game uses different versions to deploy different functionalities. Striking Legends which has 4 builds has 4 functional parts which adds different functionalities in different build according to our progress. Below we show different builds and their functionalities over other builds.

6.8.5.1 Build 1

This build served as a foundation for the game because in this build some of the most fundamental decision were made one of them was selecting the tools used in Striking Legends. Tools such as gaming engine and selecting a proper 3D modelling software because it's essential to choose the correct tools which is best suited for Striking Legends. Choosing a best suited game engine was critical because Striking Legends needed to be

an online multiplayer game. After the selection of proper tools we needed to learn and use these tools in order to implement Striking Legends and make the 3D models we needed for Striking Legends such as weapons, scenes and characters. The result of our learning formed our first build which contains the following:

- This build was all about understanding how unity and its different features work
- First, we created a dummy environment and few games objects
- We looked into how scripts are written and how can we implement an FPS controller system using unity and what are the possible ways to create it.
- After understanding the basics, we added player movements to the block character we made in blender.
- Player movement was done along with the camera movement where ever the player moved the camera would follow it, we tested this in a limited environment and checked the player movement so it wouldn't't flip.
- We added few game physics into this build by adding a collider to the ground and the player and see how they interact with each other and we applied gravity so that ground and player would collide with each other instead of passing through each other.
- We also integrated Photon cloud server with unity and tested some functionality like connecting to photon server before moving to build 2.
- As we were beginners in 3D modelling w tested how blender works and made a block character using blender.
- The block character was added to build 1 by exporting it from blender into an FBX file.
- We also made a monster model and checked how sculpting works in blender.
- Using a blocked character with few animations to test.

Figure 6.3 shows the demo environment that we used for testing.

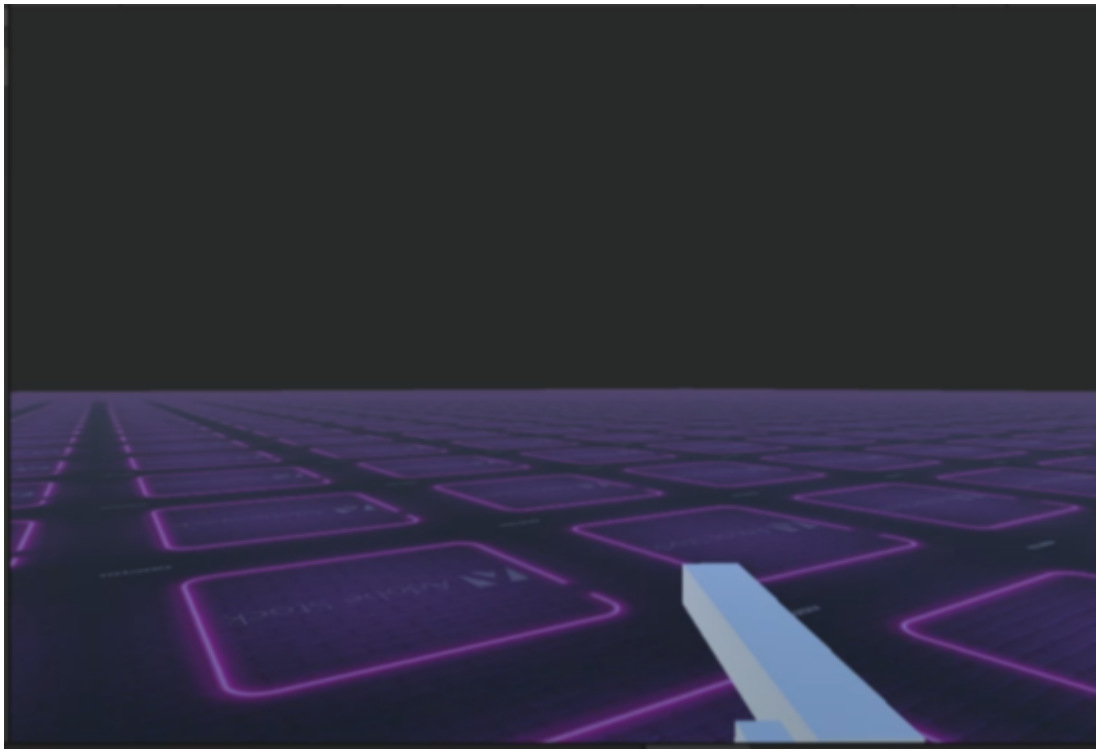

Figure 6.3: Demo Scene

6.8.5.2 Build 2

We tested this build for the followings:

- In this build we added 2 scenes to Striking Legends.
- The first scene was a menu and the 2nd scene was a game scene.
- We made a logo for Striking Legends and a background picture for our menu in PhotoShop.
- The first Scene which was a menu had the following functionalities added:
  - Connecting the player to photon server
  - A button for creating rooms
  - A button for joining random rooms
  - A button for searching available rooms
- In the second Scene (Game scene) we created a new environment/map which had a portal, walls, staircases, and few other objects and applied texture to them for realistic feel.

- We created a second character with added some animations to the player such as jumping animation, movements animations and idle animation (when the player does not move this is when idle animation is applied).
- We added many objects in the second scene so we could check the following game physics:
  - Jumping on the ground and checking the gravity physics
  - Stair case climbing
  - Slope walk
  - Collision with different game objects such as walls and other game objects
- We tested how will a player teleport from one place to another using portal we had some progress but it was not complete.
- We successfully loaded the game scene onto the network.
- We tested the camera movement along with the player movement in the network game Scene.
- We synchronized movements of players on the network so that each player movements are synchronized and visible to other players in the room.
- In blender we created the second character and rigged the character so that we can apply animations.
- We added few animations to the character such as jumping animation, idle animation, walking animations.
- We exported these animations to unity in FBX file so that we can run some tests on this character

Figure 6.4 shows the map that we used for build 2.

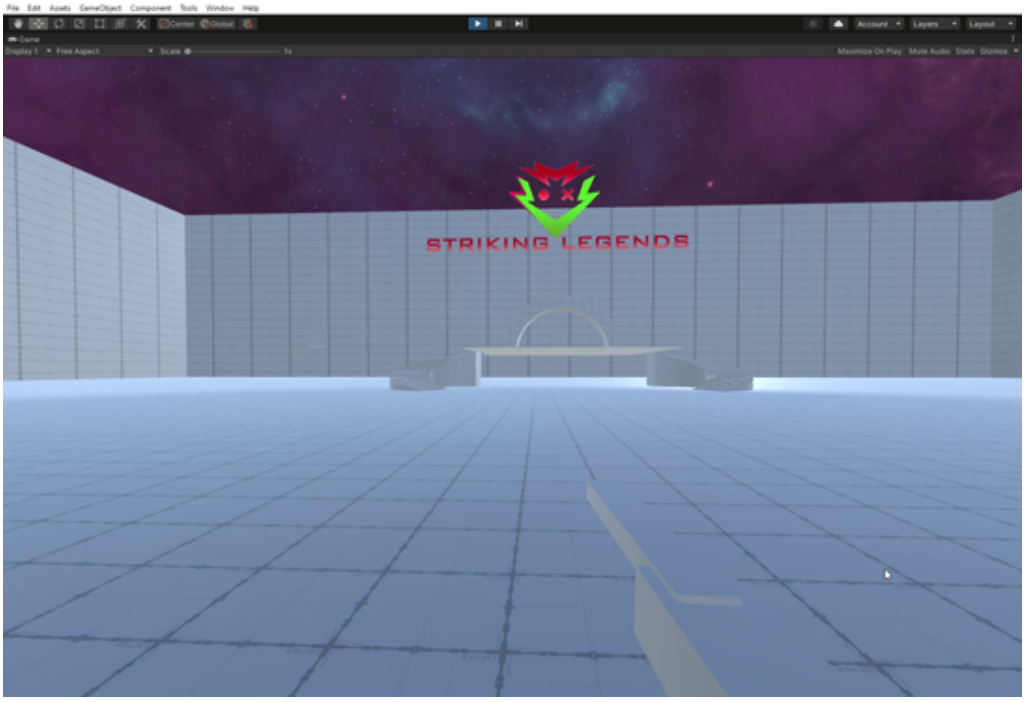

Figure 6.4: Environment Test 1

6.8.5.3 Build 3

The following functionalities were added over build 2 to build 3:

- We added 3 weapons in build 3
  - 1R-50
  - MG-7
  - P Ranger
- We added different damage points to both of these weapons and synchronizing these damage points across the network to all the players.
- We made a system for changing weapons using numeric keys and mouse wheel.
- We synchronized changing weapons across the network so that other players could see weapon changing by broadcasting this information to all the players in the room.
- We added crouch animation when a player is crouching, the player Isn't able to change weapons, player movements animation with Rifles and Pistols.
- Synchronizing all of these newly added animation across the network.
- We also tested bullet impact game physic.

- Added Aim system with a cross hair.
- Added a health system and the ability to regenerate after few seconds and synchronized this ability across the network.
- We also added teleportation ability using portal.
- Added spawn points where players can spawn in the arena by specifying few spawn points in the arena.
- Added the ability for closing the portal after 10 seconds of spawning and the ability to regenerate 2 times faster when a player used the portal for 10 seconds.
- Added lag compensation so that player can play game smoothly without network lag.
- A completely new arena like an open office with rooms and walls and other objects.
- In blender we created 1R-50 and MG-7 and a P-Ranger and we made few animations for the character we previously built and exported the animation and weapons to unity in FBX.

Figure 6.5 shows the map that we used for build 3 and figure 6.6 shows the main menu used in build 2 and build 3 and figure 6.34 shows create room menu.

Figure 6.5: Environment Test 2

Figure 6.6: Menu

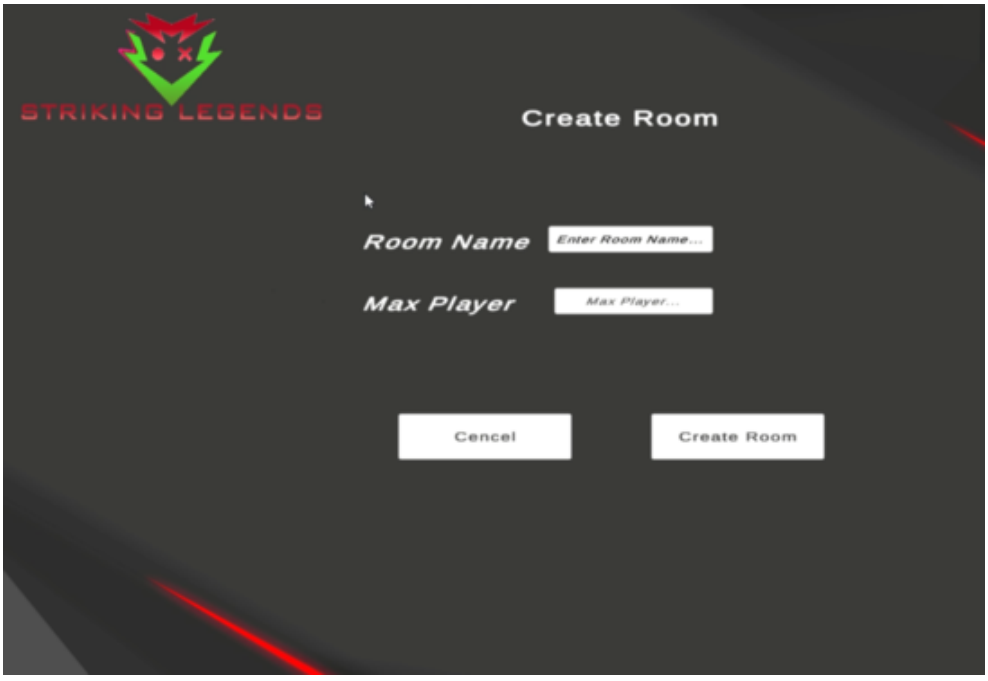

Figure 6.7: Create Room

6.8.5.4 Build 4

build 4 has the following functionalities

- A completely new menu scene

- Added 2 more weapons
  - Mega-Buster
  - P915
- Integrated playFab with unity playFab is a real-time cloud server.
- Created a player management system
- A login/sign up added in the menu scene new players can sign up and sign in no one is allowed to enter game without having an ID.
- We added skill-based matchmaking system which previously was not skill-based.
- We added points to identify user level and make skill-based matchmaking based on points.
- Points are given when a player newly creates account and when a player kill other player in room.
- Added the main character which resembles on of the developer face.
- Added few other animations with total of 14 animations included in the game.
- Added the ability to respawn after a player dies because it's a deathmatch.
- Added a timer when the time is finished the room will be closed and the player will be redirected to menu scene using a return button.
- Added the 2nd arena/map which is connected to the first map using a portal.
- Updating user point using his kills using cloud scripts.
- Leaving room option inside game scene

6.9 Weapon Models

Before going into details, we would like to discuss UV-mapping / UV unwrapping because every weapon model enlisted below has a UV-map attached to it.

6.9.1 UV-mapping/UV-unwrapping

UV-mapping is the process of projecting 2D images onto 3D objects. This process is done by unfolding the 3D model at seams to lay it out flat on a 2D surface. Seams are those parts of a mesh which is split in order to convert the 3D objects into a UV-map. U and V stands for the axes of 2D space horizontal and vertical because 2D objects has 2 axes while 3D objects have X, Y and Z axes. The process of creating a UV-map is called UV-unwrapping. Uv-unwrapping is done when the polygonal mesh is complete in order to apply texture to a 3D object to make it look realistic and good. The concept of UV-overlapping is done when we have two or more polygons on top of each other this is sometimes done to keep the texture size down in order to make it run smoother in game engines. Overlapping is specially done for weaker devices such as mobiles to boost its performance. Game engines uses these UV-maps for light baking. Figure 6.8 shows the UV-mapping process of a cube.

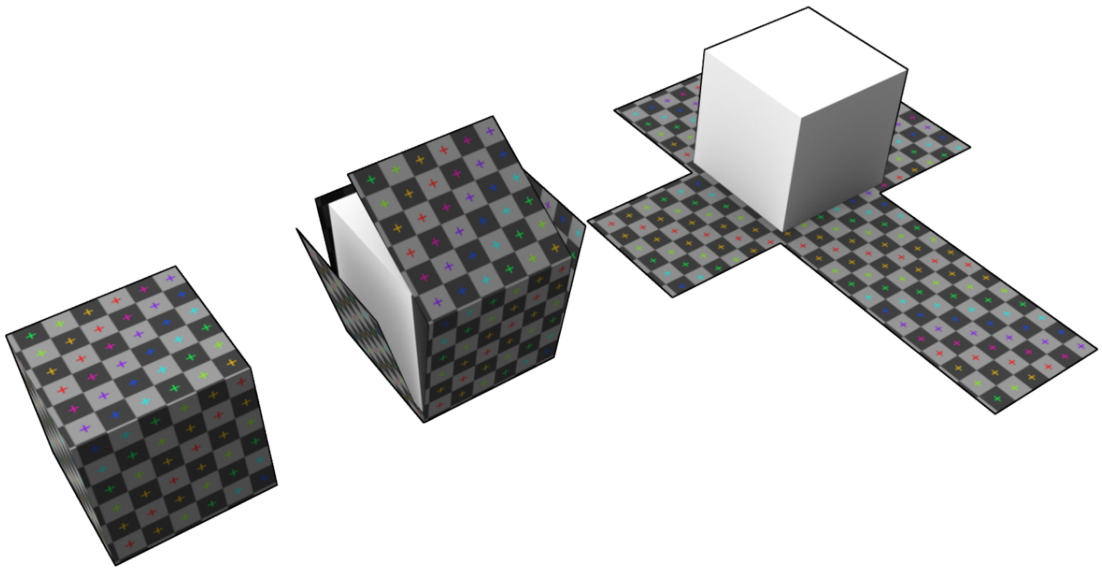

Figure 6.8: UV-map

6.9.2 1R-50 and its UV-map

Figure 6.9 shows model of 1R-50 and figure 6.10 shows the UV-map of 1R-50.

Figure 6.9: 1R-50

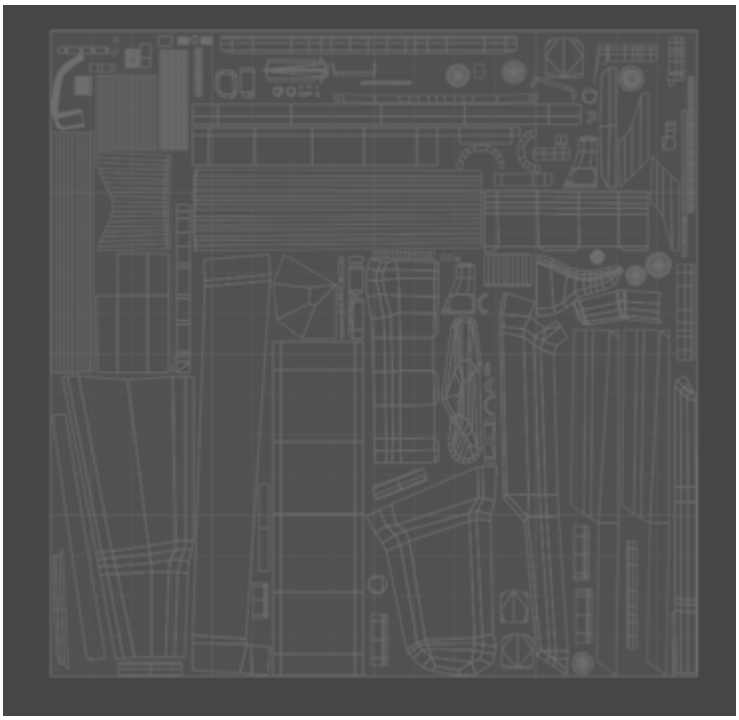

Figure 6.10: 1R-50-UV

6.9.3 MG-7 and its UV-map

Figure 6.11 shows model of MG-7 and figure 6.12 shows the UV-map of MG-7.

Figure 6.11: MG-7

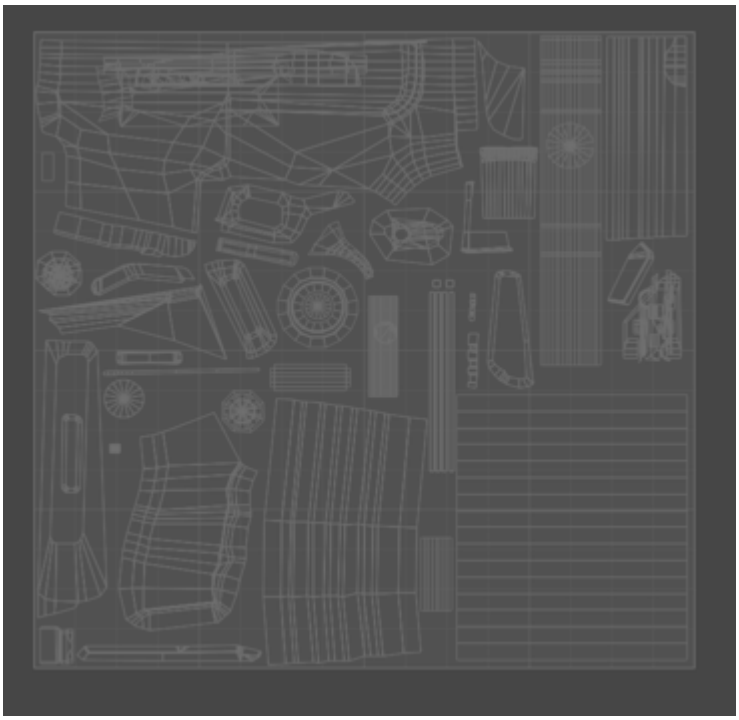

Figure 6.12: MG-7 UV

6.9.4 P-Ranger and its UV-map

Figure 6.13 shows the model of P-Ranger and figure 6.14 shows the UV-map of P-Ranger.

Figure 6.13: P-Ranger

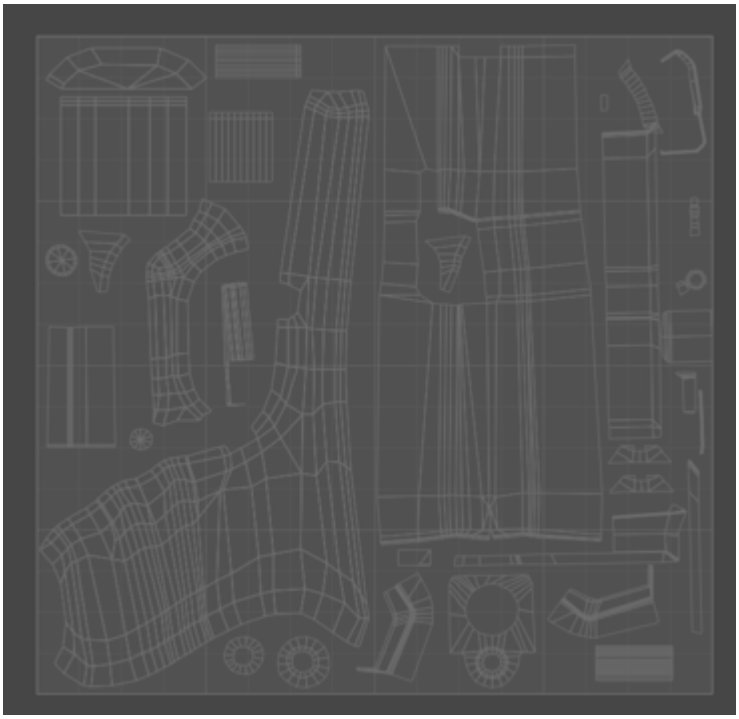

Figure 6.14: P-Ranger UV

6.9.5 Mega-Buster and its UV-map

Figure 6.15 shows the model of Mega-Buster and figure 6.16 shows the UV-map of Mega-Buster.

Figure 6.15: Mega-Buster

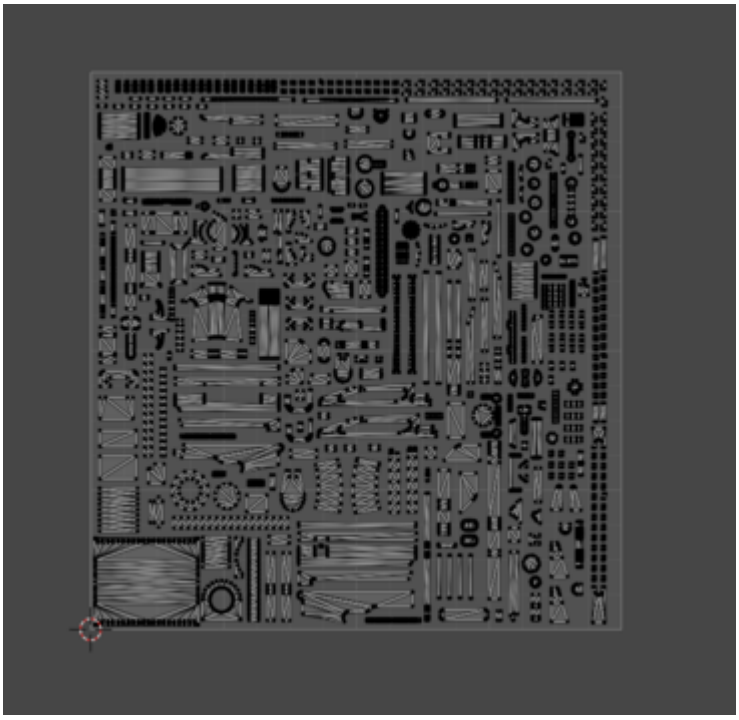

Figure 6.16: Mega-Buster UV

6.9.6 P915 and its UV-map

Figure 6.17 shows the model of Pistol P915 and figure 6.18 shows the UV-map of Pistol P915.

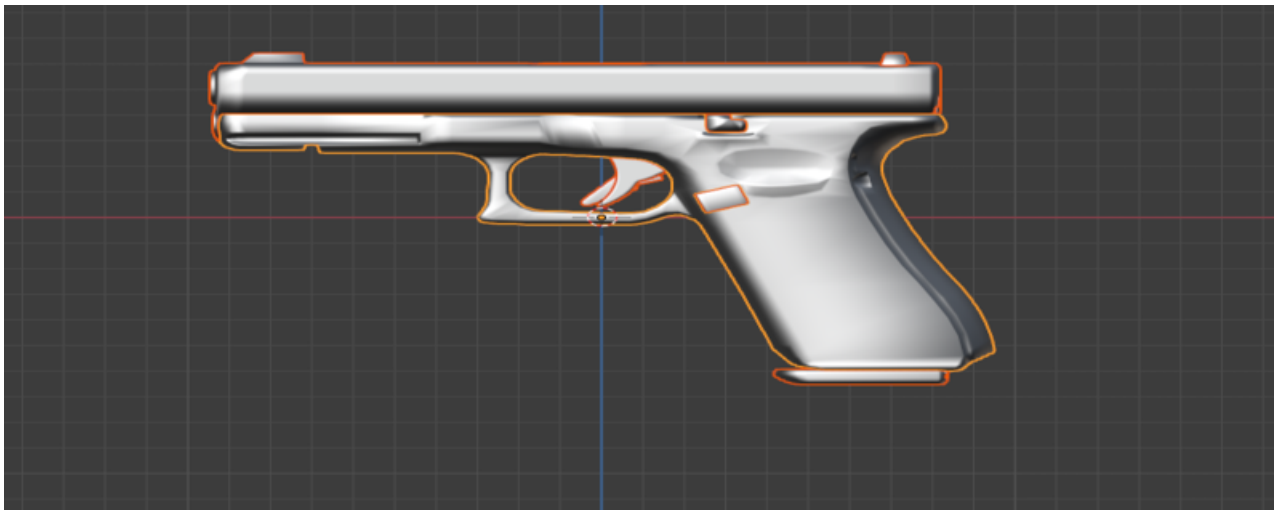

Figure 6.17: P915

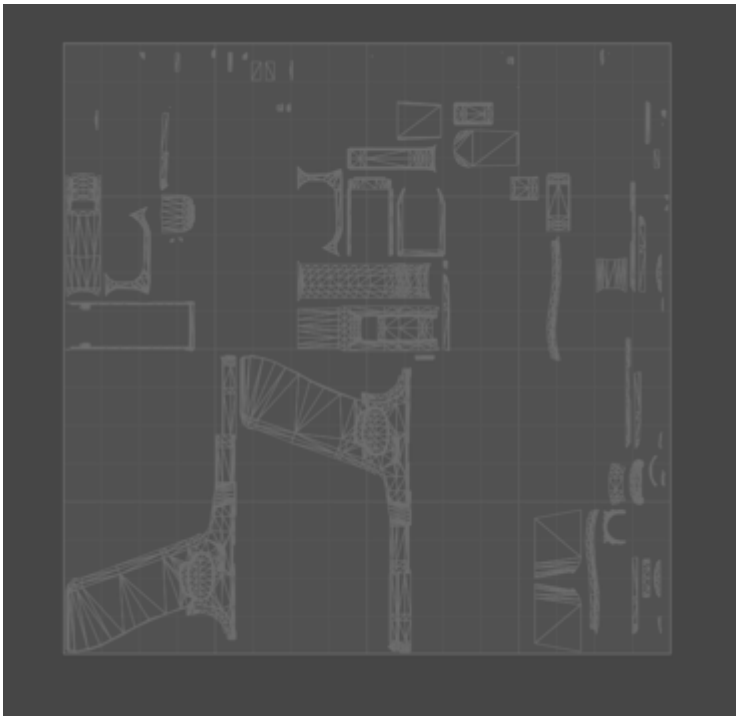

Figure 6.18: P915 UV

6.10 Scenes

6.10.1 Map One:

Figure 6.19 shows the first map of Striking Legends.

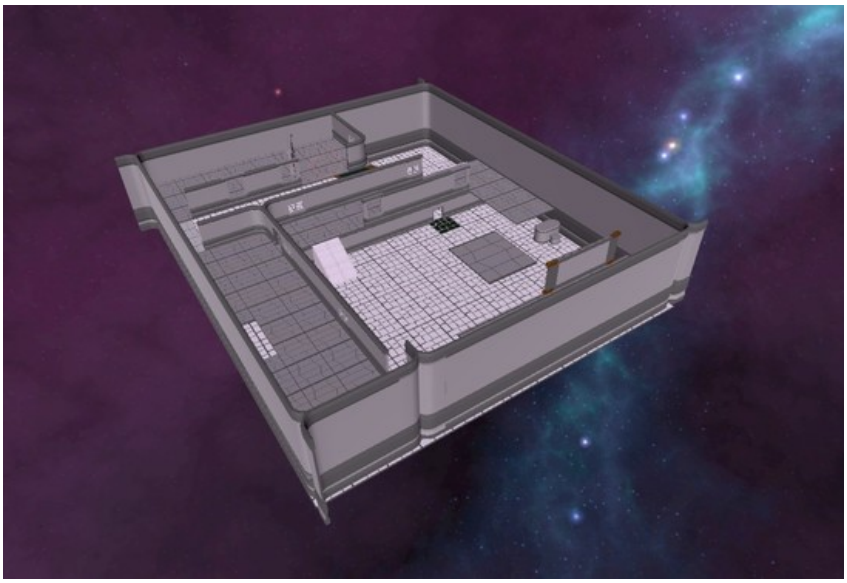

Figure 6.19: Map One

6.10.2 Map Two:

Figure 6.20 shows the second map of Striking Legends.

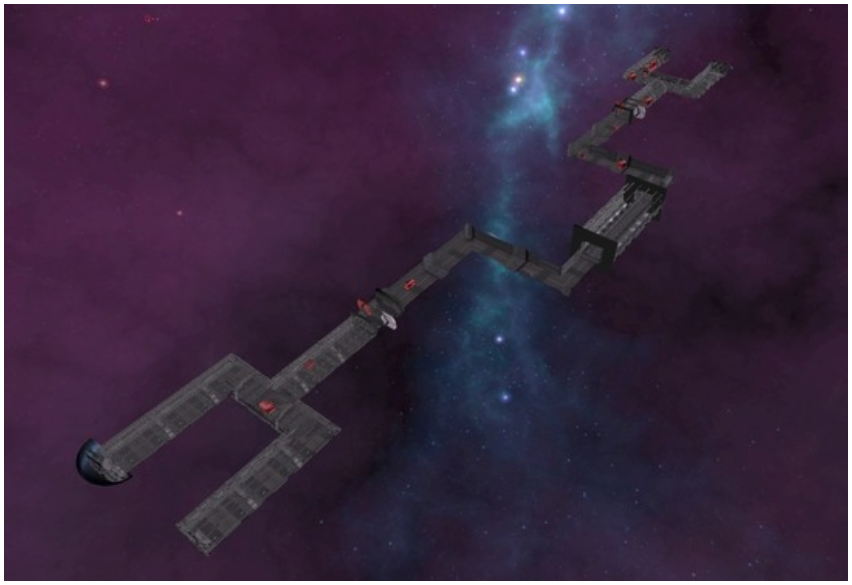

Figure 6.20: Map Two

6.11 Issues

Never has a software building been an easy task especially when it is a complex online multiplayer game like Striking Legends which has so many tools and complex implementation of idea involved in it. Through out the development of Striking Legends we came across multiple problems and issues along the way. Some of these issues were negligible but other are worth mentioning. Here we will go through some of these issues.

1. The first issue was synchronizing the players across the network which included synchronizing the animations, health and death synchronization, weapons switching synchronization. The weapon synchronization was a extremely hard and we were stuck in this problem for more than 3 weeks. To solve the problem we joined multiple Discord channels and asked many experts about the issue until someone suggested us to start over using a new method.

2. Leaving the room and returning to lobby successfully was the second major problem because player couldn't leave the room when the game was finished or during the game. We faced the following issues when returning to lobby:

    - When the master clients left the room, the room was automatically destroyed even though there were players inside the room.
    - The clients left the room everyone except the master client automatically exited the room.
    - When the player left the room, his instances were double when joining the room next time.

3. Integrating photon with playFab raised a new issued through which one couldn't join the room using the same device which we solved later on because playFab was identifying user based on their device unique identifiers.

There were many other issues that we faced but these were the major ones.

6.12 User Interface

To make it user-friendly we have used a simple UI, based on different HCI mechanism to make the usability of the game easy and interactive by providing a main menu where players can choose from 5 different options:

1. Create Room button.

2. Join Random Room button.

3. Join Available room button.

4. Check player Stats button.

5. Exit button.

6.12.1 Characters

Making character is one of the most difficult, time consuming task. In games characters are made by artist by bringing their thoughts,imaginations to physical objects in form of 3D models. Most games make generic models of main character, enemy character, robots, monsters and other game objects but there are few games which make models such that it resemble someone real face. such as making of a character which face resembles a celebrity and others. In Striking Legends, we are making the player model whose face will resemble one of the developer's face. So far no one has developed their own character which resembles the developer's face in City university So, we are the first group who has developed their own model which resemble them and put it in Striking Legends instead of using already built models.

The process of making this model was not easy because we have to go through a lot of difficulties and learning. To build such a model one has to be a master in using blender because our model was developed using Blender. In order to make this character first we needed to learn Blender , and see how different features of blender works. One of the skill we had to learn for making this character was sculpting in order to master sculpting we built some other models and a monster to develop our sculpting skill before applying it to our main model. Learning sculpting wasn't enough because when we sculpt a character the number of polygons are in millions so we have to reduce the polygons to few thousands because game engines can't render a model which has millions polygons and we can't apply animations to such models as well so, to reduce the polygons from millions to thousands we needed to do re-topology which takes a lot of times because it's done manually. The process of Re-topology is described above in Character making process.

When a character is re-topologized it loses the details because of the reduction in polygons so to make up for that we baked this low poly model onto the high poly model to make it as good as the high poly. Applying a texture is also a time consuming part because applying texture needs to 3D model needs un-wrapping them into 2D flat surface and then applying the texture. The details for UV-unwrapping is also discussed above in weapon model section. After doing all of these steps we needed animations to apply to

our character so, for that we have to rig our model which is making a skeleton called armature and then applying animations on that rigged model.
We have developed Three different characters throughout the developments phase but we are actually using only one model which resembles one of developer face that's Manzoor

6.12.2 Basic block character

Figure 6.21 shows a basic block character initially used for testing.

Figure 6.21: Basic Block Character

Figure 6.22: Detailed Block Character

6.12.3 Detailed Block Character

Figure 6.22 shows a more detailed blocked character which was used for creating animations until our own model was Created.

6.12.4 Main Character

Figure 6.23 shows the main character used in the game for the final build. The final build only contains the main Character which is made after the face of Manzoor one of the developer. This character is used by every player in the game. So, there is no separate characters for enemy players.

Figure 6.23: Main Character

6.12.5 Main Character Jacket and Jacket's UV-map

Figure 6.24 shows the jacket of the main character and figure 6.25 shows the UV-map of the jacket.

Figure 6.24: Character Jacket

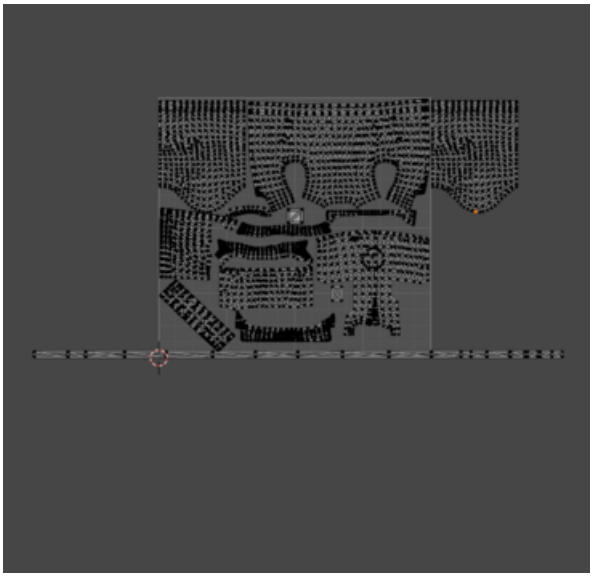

Figure 6.25: Shirt UV

6.12.6 Main Character Pants and Boots and their UV-map

Figure 6.26 shows the pant of the main character and figure 6.27 shows the UV-map of the pant.

Figure 6.26: Pants with Boots

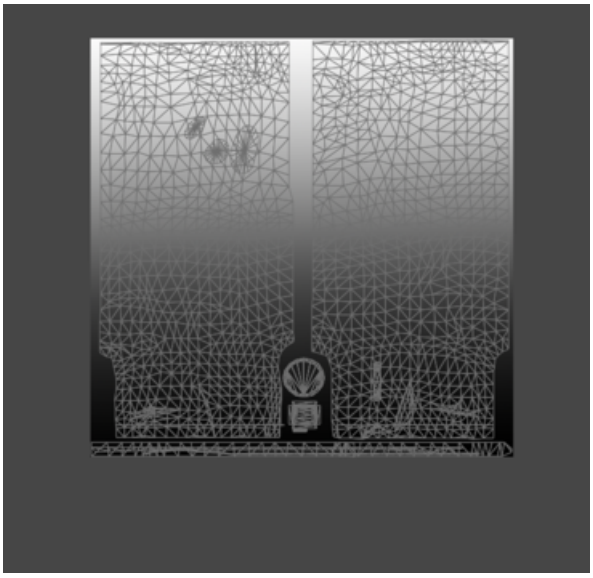

Figure 6.27: Feet UV

Seams are marked where they are not visible. Marking seams for a foot is done by selecting the edges under the feet and unwrapping them and for pants, it's done by marking the seams by selecting the side edges of the pants where the seams can't be spotted easily.

6.12.7 Main Character Hands and its UV-map

Figure 6.28 shows the hands of the main character and figure 6.29 shows the UV map of the hands.

Seams are marked by selecting edges on different fingers and selecting the edges on the sides of a hand to unwrap it.

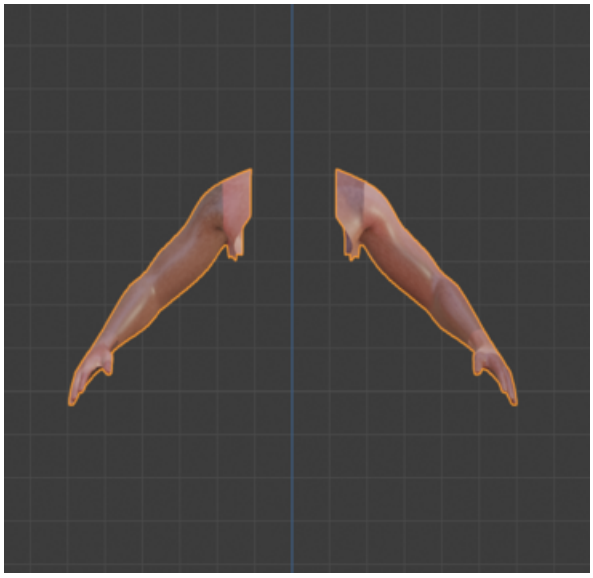

Figure 6.28: Hands

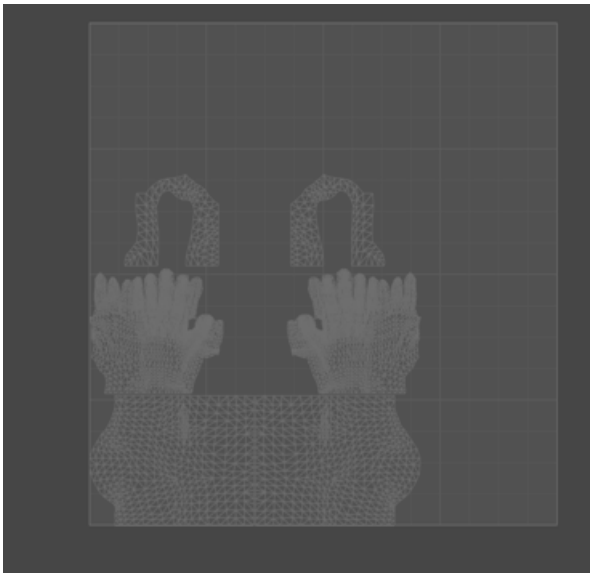

Figure 6.29: Hands UV

6.12.8 Main Character Head with its UV-map:

Figure 6.30 shows the head of the main character and figure 6.31 shows the UV-map for the head.

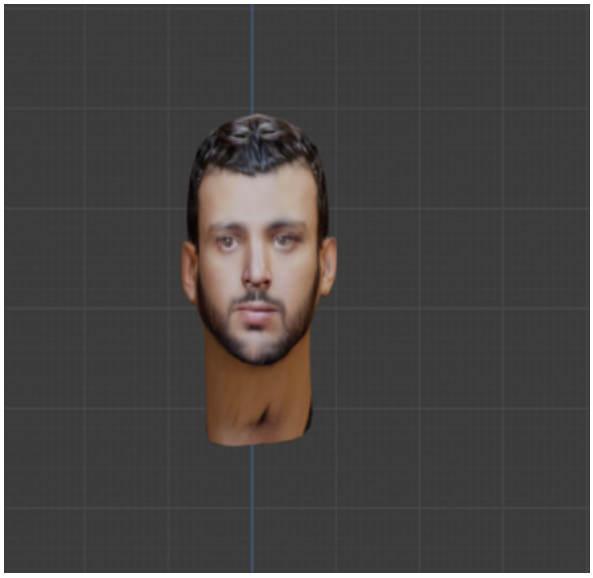

Figure 6.30: Head

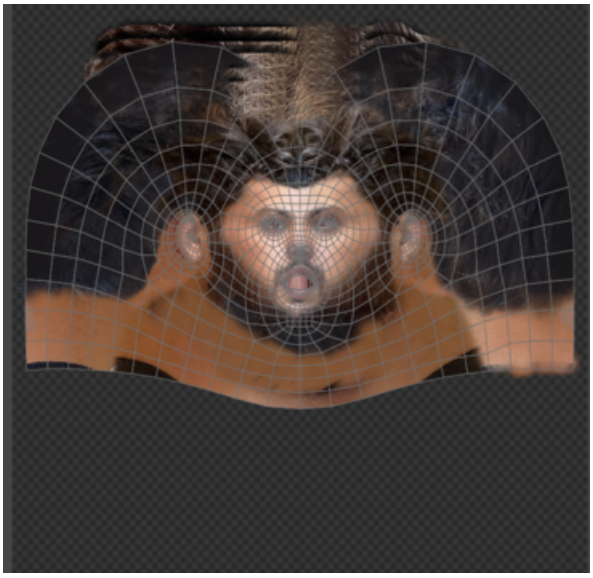

Figure 6.31: Head UV

This UV-map is called butterfly UV-map because the UV-map is shaped like butterfly. The seams are marked in the middle of head to separate them in two and a seam on the forehead and a seam under the chin area.

6.12.9 Menus

The main menu of the game has a unique design which fits the gaming theme. The main menu includes 5 buttons for 5 different tasks and a welcome message for joining players. Figure 6.32 shows the main menu interface used in build 4.

Figure 6.32: Main Menu

1. Login/sign up page
2. Create Room
3. Join random room
4. Show room list
5. Player stats
6. Exit

6.12.9.1 Login/Sign-up page

- Login, sign-up and reset password options.
- Email and password are used for login.
- Password can't be less than 6 keywords.
- Sign-up is also done using password and email.
- The user data is store in PlayFab a real-time gaming server owned by Microsoft.
- Verification of the password and email address are done using PlayFab.
- If a user forgets his password, he can reset his password.
- The link of the password is sent to the user email.

Figure 6.33 shows the login and signup interface used in build 4.

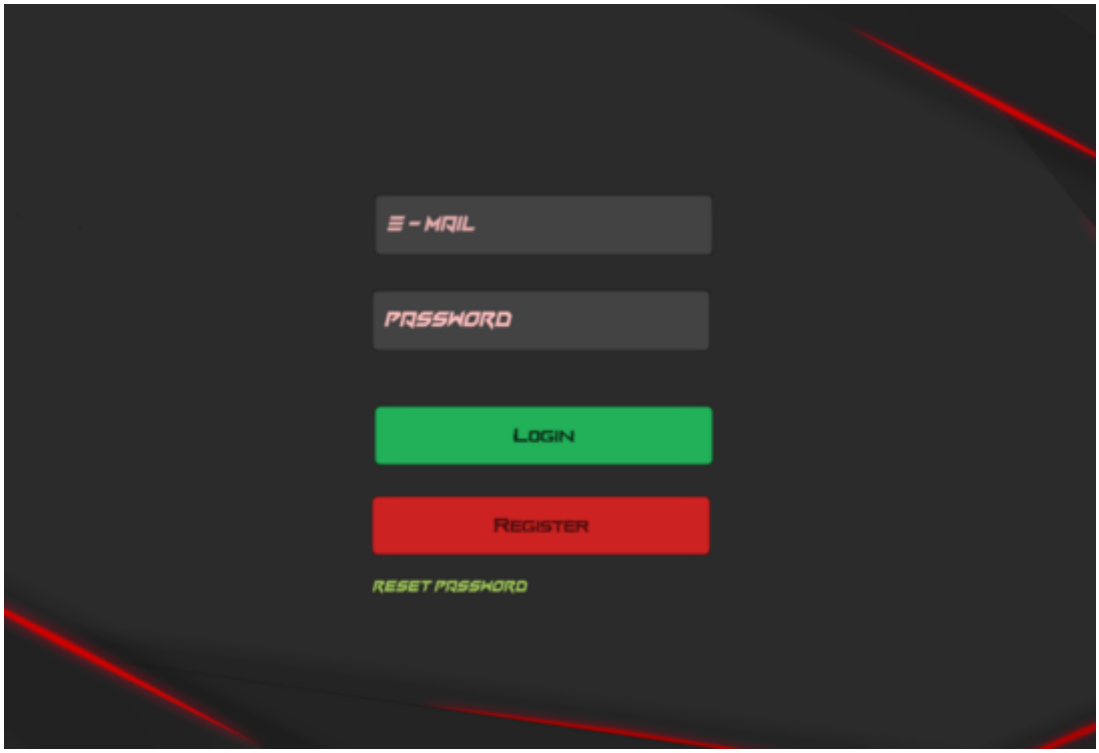

Figure 6.33: Login

6.12.9.2 Create Room

The create room button allows to create a room which has a navigation option to another menu called game creation menu. Create room button serves as an onClick event handler which directs the player to game creation menu.

- Has an input field for allowing maximum players in a room.
- Cancel button used for going back to the main menu.
- And create room button which creates the room.

Figure 6.34 shows the create room interface used in build 4.

Figure 6.34: Create Room

6.12.9.3 Joining Random Room

- Join random room has an inside menu where players search for available rooms until a match is found based on player rank.
- When a match is not found the player creates his own room where he is the master client and others can join his room if their rank matches.
- Leave button is used for leaving the room

- If the player is master-client then the start game is available to him if he randomly joins others rooms then he can't start the game until the master client starts it.
- The list shows other players who have joined you.

Figure 6.35 shows the inside room interface used in build 4.

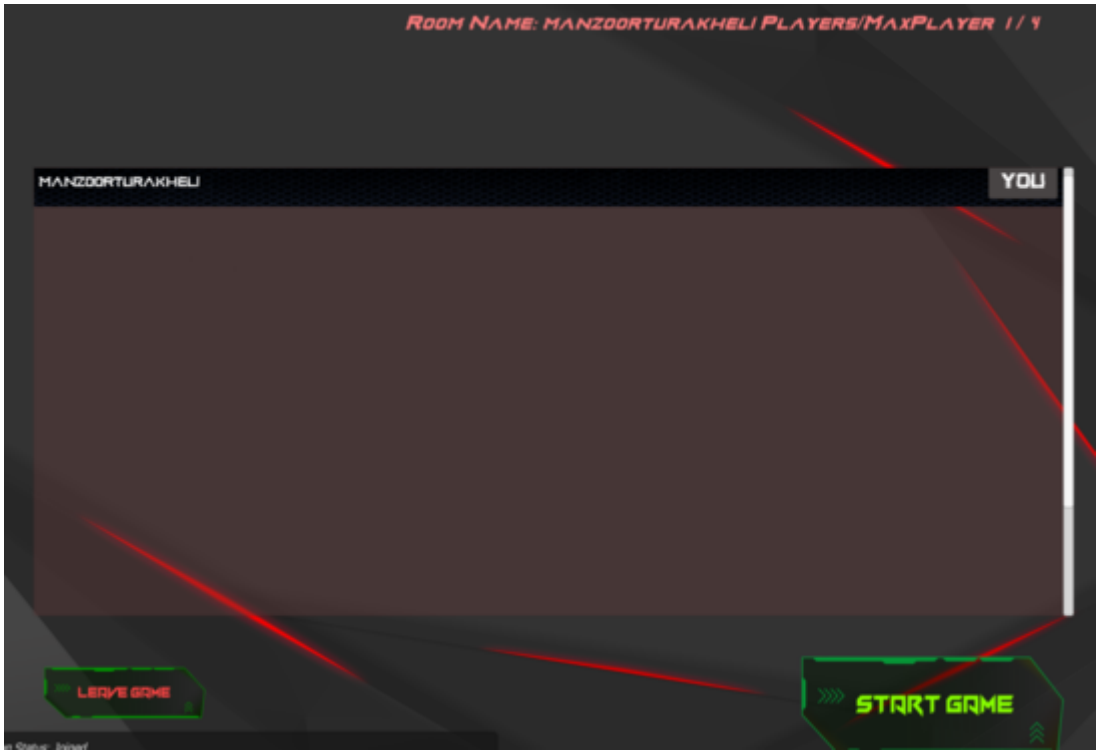

Figure 6.35: Inside Room

6.12.9.4 Show Room List

- Shows the available rooms where the player can join.
- Back button used for going back to the main menu.

Figure 6.36 shows the show room list interface used in build 4.

6.12.9.5 Player Status

- Shows player ID and his points which are used for skill-based matchmaking.
- Points are decided based on kills.

Figure 6.37 shows the show player info interface used in build 4.

Figure 6.36: Show Room List

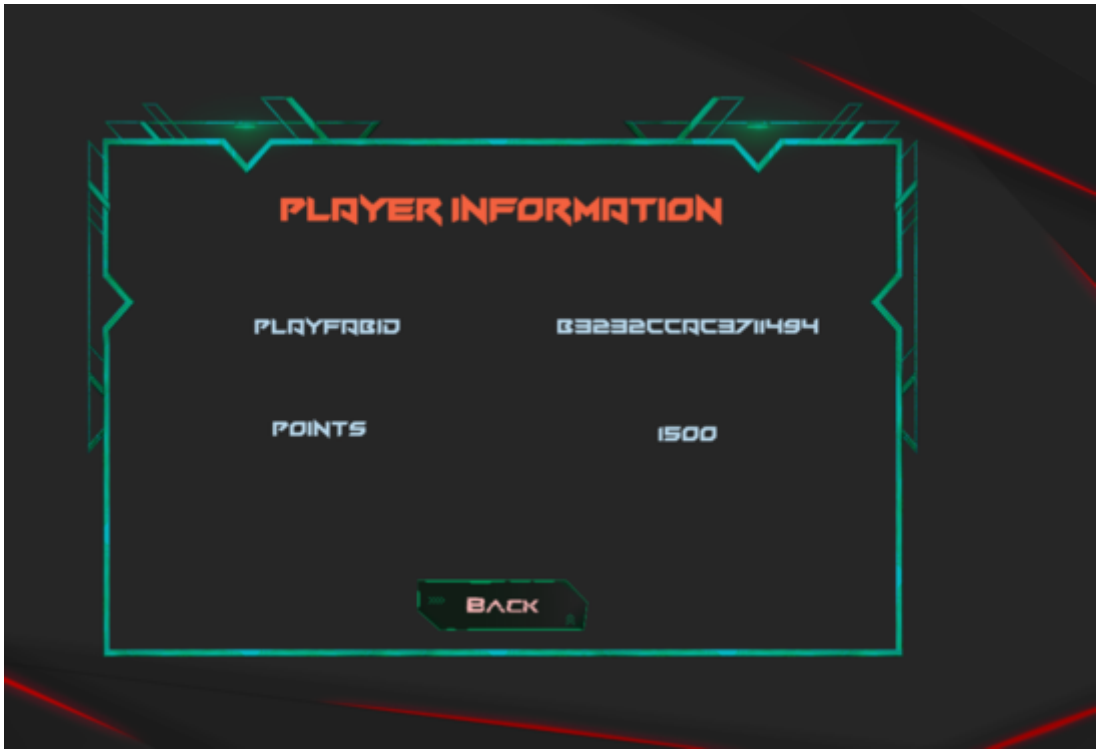

Figure 6.37: Player info

Chapter 7

System Testing

This chapter describes all the required test cases for testing and evaluating the project to make the quality of the game better by making it bug free or minimizing the bugs as much as possible. Here we will provide proper documentation of test plans for our game Striking Legends which will help us in achieving our goal quality and validate the game according to our requirements.

7.1 Purpose

Purpose of this chapter is to make proper test cases and guidelines for testing our game so that it can help us in eliminating the bugs and errors in the game. Making it bug free makes the quality better and its installation and playing easy for end users. Here we will list all the test cases and their results.

7.2 Test Approach

Our approach for testing is making different types of test cases and running them on our game Striking Legends and gathering its results and guaranteeing its success to make it a bug free game in every component whether it's on network or menu or scenes. Games usually are tested by the developers themselves during the development of the game by playing the game itself and trying to find out different bugs by cheeking the newly added functionality and seeing it whether the added functionality correctly works or not and by checking some unusual functionality such as using two functionality at the same time, for example, reloading the gun and Firing at the same time. This is done in order to test the limits of the game. This whole process is done in Unity. In unity, we can play the game for checking if any errors exist but since our game is multiplayer it should be tested on two machines or multiple builds to check whether the functionality in both builds works the same and whether the synchronization is correctly working or not. Unity has the feature to make different types of builds one of them is development build where the developer can check for errors in the build. So, we have used the development build in order to check for errors and bugs in the build. For every new functionality or editing

even a single line of code we needed to make another build which took a lot of time. In some cases compiling a build even took 16 minutes. The compilation time for a build varies because of the assets it needs to compile in order to make a build.

7.3 Test Cases

7.3.1 Sign up test

This test is done in order to check the functionality of adding new users to the player management system. Sign up is done by filling in two fields Email and Password. We have used these two fields to make the process of signing up easier because users avoid lengthy process of joining a platform; therefore we used these two fields to simplify the process. Table 7.1 shows the signup test case.

Table 7.1: Sign up test

Test Case ID: A.1.1
Pre-condition: Internet connection required because the game is multiplayer.
Steps: **1)** Open the game. **2)** You will be directed to sign up page. **3)** Fill up the fields. **4)** Click on sign up button.
Purpose: To have records of each player.
Result: Pass

7.3.2 Login test

Login test is done in order to make sure that the users who have signed up can play our game. The process of logging is also easy because users can sign in using Email and Password. The system also provides feedback in case if something goes wrong. Table 7.2 shows the login test case.

Table 7.2: Login test

Test Case ID: A.1.2
Pre-condition: Internet connection required because the game is multiplayer.
Steps: **1)** Open the game. **2)** You will be directed to sign in. **3)** Fill up the fields. **4)** Click on sign in button.
Purpose: To keep record of each player.
Result: Pass

7.3.3 Walking Animations test

In this test we check for player movement when the movement keys (WASD) are pressed and also to check the idle animation of the character when it is in standing still position. Table 7.3 shows the walking animation test case.

Table 7.3: Walking Animation test

Test Case ID: A.1.3
Pre-condition: Login is required.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Check whether idle animation is applied or not. **6)** Check the walking animations in the gameplay by pressing the required keys for the animation to apply.
Purpose: To keep the walking animation run smoothly without error.
Result: Pass

7.3.4 Shooting Animations test

Shooting animation test is done in order to check whether a weapon shakes when a player is shooting from a weapon. The shooting animation differs from pistols to rifles and it differs when the weapon is in auto or single-mode. Table 7.4 shows the shooting animation test case.

Table 7.4: Shooting Animation test

Test Case ID: A.1.4
Pre-condition: Login is required.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Check the shooting animations in the gameplay.
Purpose: To check the shooting animation running without errors because games are all about animations.
Result: Pass

7.3.5 Reloading Animations test

The Reloading animation test is done in order to check when a player presses R the reloading animation works or not and to check the auto reloading animation when a weapon is out of bullets. Table 7.5 shows the reloading animation test case.

Table 7.5: Reloading Animation test

Test Case ID: A.1.5
Pre-condition: Login is required.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Check the Reloading animations in the gameplay.
Purpose: To keep the animation running smoothly without error.
Result: Pass

7.3.6 Crouching Animations test

The Crouching animation test is done in order to check whether crouching animation works when holding the C button. This animation is applied to the character as long as the player holds the C button. Table 7.6 shows the crouching animation test case.

Table 7.6: Crouching Animation test

Test Case ID: A.1.6
Pre-condition: Login is required.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Check the Crouching animations in the gameplay.
Purpose: To check whether crouching animation works correctly.
Result: Pass

7.3.7 Health test

Testing health is a vital part of our game because our game is all about kills, deaths and regeneration of health. So we needed a proper mechanism to test these. Table 7.7 describes the health test case.

Table 7.7: Health test

Test Case ID: A.1.8
Pre-condition: Must be inside the game scene.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up page. **3)** In menu click on create room. **4)** Now start the game. **5)** Fire your enemy. **6)** Check whether your health decreases or not (health is 100). **7)** After 5 seconds check whether your health regenerates or not.
Purpose: To have a fully functional damage and health regeneration system.
Result: Pass

7.3.8 Damage test

To kill someone damage test is required. This test checks how different weapons have different damage points. This test is done in order to give damage to other players and eventually die after taking 100HP damage before regeneration of health. Table 7.3.8 describes the damage test case.

Table 7.8: Damage test

Test Case ID: A.1.9
Pre-condition: Must be inside the game.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Use different weapons to attack and fire your enemy or yourself. **6)** Check whether each weapon has its own damage points.
Purpose: To have a fully functional damage and attack system.
Result: Pass

7.3.9 Player controller test

Player Controller test is the test where a player can check the controlling system of the game.This test describes proper guidance for checking and testing the player controller system. Table 7.3.9 describes the player controller test case.

Table 7.9: Player Controller test

Test Case ID: A.1.10
Pre-condition: Must be inside the game.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Check WASD for different direction movement. **6)** Check mouse buttons for fire and weapon changing. **7)** Check 1 to 5 numeric pad values for weapon changing. **8)** Check Space button for jumping , shift+ W for faster.
Purpose: To have a fully working player controller system.
Result: Pass

7.3.10 Spawning test

Spawning test is done in order to check whether a player respawns after dying and checking whether the player respawns to different locations defined in spawn points. Table 7.10 describes the player spawn test case.

Table 7.10: Player Spawn test

Test Case ID: A.1.11
Pre-condition: Must be inside the game
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Check where you are spawned. **6)** Get killed to check whether respawning works. **7)** If it works check are you respawned to a different spawning points.
Purpose: To have a respawning system where player is respawned when killed to different point in the game arena.
Result: Pass

7.3.11 Physics test

Physics test is done in order to check the different types of physics in our game such as checking the collision of player and a bullet, collision with different types of objects in the game. Table 7.11describes a test case to check game physics.

Table 7.11: Physics test

Test Case ID: A.1.12
Pre-condition: Must be inside the game.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Walk and jump on the ground and make collision with walls. **6)** Check whether collision and physics works in the terrain.
Purpose: To have a full physics system.
Result: Pass

7.3.12 Synchronization test

This test case checks whether synchronization of animations,player movements, gun firing, health regeneration, taking damage , weapon reloading is all synchronized across the network. Table 7.12 describes the synchronization test case.

Table 7.12: Synchronization test

Test Case ID: A.1.13
Pre-condition: Must be inside the game.
Steps: **1)** Open the game. **2)** You will be directed to login/sign up. **3)** In menu click on create room. **4)** Now start the game. **5)** Another player should also join the same room. **6)** Check whether player movement to gun reloading is working properly or not. **7)** Now start the game.
Purpose: to have a synchronise system across the game because it's a multiplayer game.
Result: Pass

7.3.13 Skill-based Matchmaking test

Skill-based matchmaking test checks whom plays with whom. Because the match-making is process is based upon points so it should be clear that a player of high level with more points shouldn't play with a player with lower points. Table 7.13 describes the skill-based matchmaking test case.

Table 7.13: Skill-based Matchmaking test

Test Case ID: A.1.14
Pre-condition: Must be inside the game.
Steps: **1)** In menu click on create room. **2)** Now start the game. **3)** Another player should join the room randomly. **4)** Check whether the skill-based matchmaking is working Skill-based match making process is described in implementation chapter.
Purpose: To have a skill-based matchmaking system
Result: Pass

7.3.14 Portal test

Portal test is done in order to check the teleportation ability of players and to check whether a player can teleport from one map to another using the portal. Table 7.14 describes the portal test case.

Table 7.14: Portal test

Test Case ID: A.1.15
Pre-condition: Must be inside the game.
Steps: **1)** In menu click on create room. **2)** Now start the game. **3)** Go to the portal area in the first arena. **4)** Check whether the portal teleport the player to the next arena and can the player comeback by using the same portal to the first arena.
Purpose: To have a teleportation system for parallel arenas/maps.
Result: Pass

7.3.15 Full Game Run test

Full game run test is done in order to check whether every feature of the game work like how it was intended to work. Table 7.15 describes the full game run test case.

Table 7.15: Full Game Run test

Test Case ID: A.1.16
Pre-condition: To have the setup and installed it and have a working.
Steps: **1)** Open the game and check whether it opens. **2)** After opening checking the sign-up functionality whether it work. **3)** Checking the login functionality whether it works. **4)** Checking all the buttons on the main menu and clicking on all of them. **5)** Playing a game successfully and checking all the animation and synchronizing of the players. **6)** Checking the health and damage system. **7)** Checking portal. **8)** Checking everything.
Purpose: To have a fully bug free game.
Result: Pass

Conclusion

Striking Legends is a strategy game which provides amazing theme of death match multiplayer games and a different gaming experience by providing parallel arenas and a portal to switch between these arenas. Players fight real players with no AI players involved. Players can only in solo mode instead of duo or team. The fighting is done using different weapons which have different damage points. Striking Legends also motivates local Pakistani game developments industry into developing multiplayer games and helps in upbringing the Esports community of Pakistan. Striking Legends also successfully experimented the use of Evolutionary prototyping and Throw away prototyping by achieving the scopes and objectives for this project in timely manner and because this model helped us in clearing the requirements and removing the ambiguities throughout the whole development phase and also helped us in building four functional builds.

References

[1] N. Gilbert, *ßSSNumber of gamers worldwide 2021/2022: ßSSDemographics, statistics, and predictions*, https://financesonline.com/number-of-gamers-worldwide/, 2021.

[2] Wikipedia, *ßSSPc game*, https://en.wikipedia.org/wiki/PC_game, 2021.

[3] Pcgamesn, *ßSSThe best multiplayer games on pc in 2021*, https://www.pcgamesn.com/best-multiplayer-games, 2021.

[4] R. Zubek, *ßSSElements of game design*, https://mitpress.mit.edu/books/elements-game-design, August 2020.

[5] Wikipedia, *ßSSUnity*, https://en.wikipedia.org/wiki/Unity_(game_engine), 2021.

[6] A. Andrew Liptak, *ßSSHow neill blomkamp and unity are shaping the future of filmmaking with adam: ßSSThe mirror*, https://www.theverge.com/2017/10/4/16409734/unity-neill-blomkamp-oats-studios-mirror-cinemachine-short-film, November 2017.

[7] T. Roosendaal, *ßSSHow blender started, twenty years ago...* https://code.blender.org/2013/12/how-blender-started-twenty-years-ago/, December 2013.

[8] ——, *ßSSBlender's 25th birthday!* https://www.blender.org/press/blenders-25th-birthday/, January 2019.

[9] J. Allen, *ßSSWhat is photoshop?* https://www.lifewire.com/what-is-photoshop-4688397, March 2020.

[10] Wikipedia, *ßSSC sharp (programming language)*, https://en.wikipedia.org/wiki/C_Sharp_(programming_language), 2021.

[11] Photonengine, *ßSSC sharp (photon)*, https://www.photonengine.com/, 2020.

[12] E. Games, *ßSSPun2*, https://assetstore.unity.com/packages/tools/network/pun-2-free-119922, 2016.

[13] PlayFab, *ßSSPlayfab*, http://playfab.com/, 2019.