CS

COC251

B811469

**FIRST PERSON SHOOTER**

**GAME**

By

Christos Vrionides

Supervisor: Dr W. Hussak

Department of Computer Science

Loughborough University

May/June 2021

**Abstract**

The purpose of this project was to develop and First-Person Shooter (FPS) survival game application for PC and Mac users using Unity engine. The games’ purpose was to provide a fast passed game that users can play to spend their time and relax. The application was designed and tested for both platforms with little to no difference between them. For the development process of the software, small details of the Unity engine were explored and experimented with in order to maximize its potential in the project.

**Acknowledgements**

I would like to thank my project supervisor, Dr W. Hussak, for his advice and support throughout the development of this project.

Table of Contents

[1 Introduction 4](#_Toc71104901)

[1.1 Aim 4](#_Toc71104902)

[1.2 Objectives 5](#_Toc71104903)

[1.3 Project Plan 5](#_Toc71104904)

[1.3.1 Planning 5](#_Toc71104905)

[1.3.2 Gantt chart 6](#_Toc71104906)

[2 Literature Review 6](#_Toc71104907)

[2.1 Methodologies 6](#_Toc71104908)

[2.1.1 Waterfall 7](#_Toc71104909)

[2.1.2 Agile 7](#_Toc71104910)

[2.1.3 Incremental 8](#_Toc71104911)

[2.2 Development Environments 9](#_Toc71104912)

[2.2.1 Unity 10](#_Toc71104913)

[2.2.2 Unreal Engine 10](#_Toc71104914)

[3 Requirements 11](#_Toc71104915)

[3.1 Requirement specification 11](#_Toc71104916)

[3.2 Requirements table 12](#_Toc71104917)

[4 Build and Design. 16](#_Toc71104918)

[4.1 Main menu 16](#_Toc71104919)

[4.2 Game User Interface 17](#_Toc71104920)

[4.3 Enemies 18](#_Toc71104921)

[4.4 Weapons 19](#_Toc71104922)

[4.5 Game Controls 20](#_Toc71104923)

[4.6 Pause Menu 21](#_Toc71104924)

[4.7 Map Design 22](#_Toc71104925)

[4.8 Levels 24](#_Toc71104926)

[4.9 Sound 25](#_Toc71104927)

[4.10 Game End states 25](#_Toc71104928)

[5 Implementation. 26](#_Toc71104929)

[5.1 Implementation Introduction 26](#_Toc71104930)

[5.2 Learning Unity 27](#_Toc71104931)

[5.3 Main menu 27](#_Toc71104932)

[5.4 Player controls 27](#_Toc71104933)

[5.5 Enemies 28](#_Toc71104934)

[5.6 Weapons 28](#_Toc71104935)

[5.7 User Interface 29](#_Toc71104936)

[5.8 Game design 29](#_Toc71104937)

[5.9 Implementation Conclusion 30](#_Toc71104938)

[6 Testing 30](#_Toc71104939)

[6.1 Testing Description 30](#_Toc71104940)

[6.2 Unit Testing 31](#_Toc71104941)

[7 Evaluation 35](#_Toc71104942)

[7.1 Skills acquired. 35](#_Toc71104943)

[7.2 Requirements 36](#_Toc71104944)

[7.3 Future Additions 36](#_Toc71104945)

[7.4 Conclusion 36](#_Toc71104946)

[8 References 36](#_Toc71104947)

# Introduction

## Aim

Many people waste time while they are on the computer doing nothing, maybe they are waiting for something to load or something to download. Other times they could be bored and have nothing to do and are in need to do something to pass time. To combat this issue the aim of this project is to create a fast-paced First-Person Shooter survival game that is easy for the user to learn within a few minutes. The game will consist of several levels where the player needs to kill all the enemies to proceed to the next level until eventually reaching the final level and winning the game. The enemies will be following the player in order to kill him, so they player needs to move and use the maps obstacles to his advantage to avoid them.

## Objectives

To achieve the projects aims to its best possible form, the following objectives need to be satisfied:

1. Preform research into existing FPS survival games and find the best game design.
2. Research the technologies available for the implementation of the game and make an informed decision on what I will be using.
3. Specify the requirements of the game and have clear goals on what I need to do.
4. Analyze and learn how to use the software I will be using.
5. Develop the game controls that can take input from the user’s keyboard and mouse.
6. Learn how to design player, enemy and map models for the game.
7. Compile and implement everything together to have a working functional game.
8. Fully test the game and fix all bugs to determine if all the system requirements and aims have been met.
9. Create a report on the development of this project and its progression.

## Project Plan

### Planning

The plan for the project was to divide each big task into small sections where it was made easier to focus on each one. Starting from the first task, every task had to be completed before continuing to the next in order to have a uniform progression along the project. (easy to fix issues?)

### Gantt chart

A screenshot of a computer

Description automatically generated with low confidenceIn Figure 1.1 is the Gantt chart for this project that shows the progression of the work done for the 30 weeks spent on this project.

*Figure 1.1*

# Literature Review

## Methodologies

The project development methodology the project initially intended to follow was agile. But after evaluating all options for different development models as discussed below, the model used for this project was the incremental model. Since the requirements for this project was clear from the start and different functionalities were separate from each other. By using the incremental model, it was easy to make changes to a specific functionality if a change was needed, making it the most effective and efficient model for the project.

### Waterfall

The waterfall development model is a more traditional path, where requirements are solidified at the beginning of the process all the way throughout the project [1]. It consists of sequential stages where each stage must be completed in order to continue to the next (as shown in Figure 2.1).

If a project is close to be completed earlier stages in the model cannot be revisited. This development model could be considered for this project because the set of requirements are set and there is a low risk of them changing with little flexibility.

The waterfall model was not ideal for this project because of the lack of flexibility and there being changes in requirements during the project it would be difficult to fix.

A picture containing text, businesscard, screenshot

Description automatically generated

*Figure 2.1 (Waterfall method stages[2])*

### Agile

The agile model unlike the waterfall, is a more mature. It allows the project to be developed in a faster pace while small increments are released and then revisited [3]. When requirements are changed it is easy to adapt and change to the responses of a project. Agile is the perfect model for a project that is undecided or has the possibility of change on its requirements making it flexible (Figure 2.2).

Diagram

Description automatically generated

*Figure 2.2 (Agile methodology[4])*

Conclusion on Agile development for this project is, although Agile is a more sophisticated and modern development model, the requirements of the project are fixed. Therefore, agile would not have been a good choice for the project.

### Incremental

The incremental model (shown in Figure 2.3 [6]) is the process of breaking down the project into smaller increments to be delivered. Requirements of the individual increments are realized at the start of the development of each section [5]. The functionalities of each increment are added to each other thus slowly creating the application. If there is a change in requirements in a specific increment, it can always be revisited without changing other functionality.

Diagram

Description automatically generated

*Figure 2.3 (Incremental Methodology[6])*

Since the project can be divided into smaller increments like map design, game design, coding and more. Also being a possibility that there will be changes in requirements in each, an incremental approach to the project would be a suitable development model.

## Development Environments

The decision for this project was to make a game, therefore an integrated development environment need to be chosen based on what functionalities they offer and its ease of use. There are multiple different game development engines that can be used to create games. Most games

### Unity

The Unity engine is the most popular game engine in the word that has a lot of features and is flexible in order to create almost any game imaginable. Unity uses scripts that can be written in C# and its own 3D scene and animation builder. The engine was developed since 2005 and has gathered a massive following of users that contribute to its growing documentation including blogs, videos and sites filled with information.[7]

A screenshot of a computer

Description automatically generated with medium confidence

[Figure](https://unity3d.com/beta/2020.1b) *2.4(Unity IDE [8])*

### Unreal Engine

Unreal engine is like Unity, but there are some key differences. It is also a widely used game engine and is used for many modern big games likes ‘Fortnight’ and ‘Rocket League’. You can use scripts that can be written in C++ or even its own visual ‘blueprints’ to create a game for people that can’t or don’t know how to code. Even though it may seem a bit difficult at first it has powerful material and animation tools that can be used to create stunning maps and scenery once mastered.[9]

A screen shot of a video game

Description automatically generated with medium confidence

Figure 2.5(Unreal Engine IDE [10])

There are multiple game engines to create a game, but there are only two that stand out the most. The two that stuck out from the others were Unreal engine and Unity. They are similar features and are both powerful, but it has been concluded that the best engine for this project is the Unity engine since it offers a more straightforward design than the Unreal engine with a lot of functionality and much more documentation online. Also contributing to the decision of choosing Unity was the languages. There is a small difference between C++ and C# in the scope of making a script for a game but, C# is a more powerful and a simpler language to use in practice.[11]

# Requirements

## Requirement specification

The requirements for this project are defined by the following structure:

**Requirement number**: The number of the requirement in the list.

**Category**: The Category of which the requirement resides in e.g., Gameplay, Game user interface.

**Description**: A short definition of what the requirement needs to do.

**Priority**: The priority level of the requirement and what importance it has to the project using the MoSCoW prioritization method that is mostly used with projects that have a set timeline[12].

Must - These are the requirements that must be satisfied within the project.

Should - The requirements that are important to the project and should be implemented if possible.

Could - These requirements are desirable for the project but not essential.

Wont - These requirements are out of the scope of the project and will not be included.

**Rationale**: The reasoning behind the need of the requirement.

## Requirements table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Requirement number | Category | Description | Priority | Rationale |
| 1. | Start Menu | Start Menu UI should show up once application is launched | Must | The start menu UI should be the first thing a user sees when the game is launched |
| 2. | Start Menu | Quit Button should exit game | Must | Interface should allow the user to exit the game |
| 3. | Start Menu | Should have a user manual on how to play the game | Should | Allows the user to understand how to play the game |
| 4. | Start Menu | The menu should have a start button that starts the game | Must | The user must be able to start the game when he is ready by a click of a button |
| 5. | Gameplay | The user should be able to move the character with the W, A, S, D keys on their keyboard | Must | The user must move their character to play the game |
| 6. | Gameplay | The user should be able to look around when moving their mouse | Must | The user must look around to play the game |
| 7. | Gameplay | The user should be able to attack with their weapon using a mouse click | Must | The user must be able to shoot to play the game |
| 8. | Gameplay | The user should be able to switch between weapons | Must | The user needs to be able to switch weapons to choose the most ideal one in the situation he is in |
| 9. | Gameplay | The user should be able to Jump | Must | The user must jump to get over obstacles in the map |
| 10. | Game Design | Map Must be small | Should | For it to be easy to navigate |
| 11. | Game Design | Map must contain obstacles | Must | To make the map more complicated |
| 12. | Game Design | Enemies should follow player | Must | To be able they can get in range to damage the player |
| 13. | Game Design | Enemies should damage player when they get close | Must | In order to kill the player |
| 14. | Game Design | Enemies take damage when attacked by the player | Must | In order to lose health points |
| 15. | Game Design | Enemies die when they have no health left | Must | In order to proceed to the next rounds |
| 16. | Game Design | Enemies of next round are spawned after all current round enemies have died | Must | For the player to proceed to the next round |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 17. | Game Design | Enemies in each level/round get faster and stronger | Must | For the game to get harder as you progress |
| 18. | Game Design | There will be 10 rounds of enemies | Must | 10 rounds are plenty of rounds to get through |
| 19. | Game Design | Enemies should have an indicator on how much health they have | Should | For the player to know how much health points the enemies have |
| 20. | Game Design | The player must have multiple weapons | Must | So that the player has a variety of weapons to choose against the enemies |
| 21. | Game Design | Weapons must have different range, damage and fire rate | Must | For each weapon is better or worse in different situations |
| 22. | Game User Interface | Player should have a pause menu | Must | For the user has a chance to pause the game if bored or needs to take a break |
| 23. | Game User Interface | Pause menu should have a resume button | Must | For the user can resume any time he wants |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 24. | Game User Interface | Pause menu should have a quit button | Must | For the user to be able to quit the game whenever he wants |
| 25. | Game User Interface | UI should display what level the player is on | Should | For the user to know what level he is on |
| 26. | Game User Interface | UI should display the stats of the weapon equipped | Should | For the user to know what the weapon statistics are |
| 27. | Game User Interface | UI that shows the players current health | Must | For the user to know how much health he has |
| 28. | Game User Interface | If player dies a Game Over screen shows | Must | To know that the game is over |
| 29. | Game User Interface | If the player wins all rounds a Game Won screen shows | Must | For the user to know he won the game |

# Build and Design.

The game was designed as to a good standard following almost all the requirements. The style of the game was made to look simplistic and blocky. Once the game has been installed to the computer, when opening the application, the first thing the user will see is the Main menu.

## Main menu

In the main menu there are 3 buttons that can be pressed. (figure 4.1)

**Play button**: When clicked, the game starts.

**How to play button**: Will open a page where it will explain how to play the game and what the goals of the game are (as shown in figure 4.2)

**Logo

Description automatically generatedQuit**: When clicked it will exit the application.

Text

Description automatically generated*Figure 4.1 (Main Menu)*

*Figure 4.2 (How to play page)*

## Game User Interface

The games user interface is what the player will see on his screen when the game has started. The player can see different types of information anytime in the game. The user interface consists of:

**Health bar**: As shown in the bottom-middle of Figure 4.3 is the current health the player has.

**Weapon**: Is the weapon equipped at the time by the player shown on Figure 4.3 bottom-right of the screenshot.

**Weapon Details**: Can be seen at the bottom-left of the user’s screen.

**Level**: The current level the player is in can be seen at the top middle of the screen.

Graphical user interface

Description automatically generated

*Figure 4.3(Health bar/Weapon/Weapon details/Level)*

## Enemies

The enemies are the ‘bad guys’ of the game. And their objective is to get close to you to deal damage.

**Design**: The enemies are the blocklike creatures (as show in Figure 4.4).

**Health** **bar**: The enemy health bar is the green bar above the enemy heads that indicates the enemy’s current health.

**Movement**: The enemies need to move in order to get close to the player and deal damage. So that the enemies find the most ideal path to get close to the player.

**A screenshot of a video game

Description automatically generated with medium confidenceEnemy types**: There are two enemy types. The normal one that has a white color, and the boss version that has a red color (as shown in Figure 4.5). The normal ones get faster and stronger in each round, but the boss can only be found once in the final level and has ten times the health of a normal enemy.

*Figure 4.4(Enemies in the map)*

A picture containing graphical user interface

Description automatically generated

*Figure 4.5(Normal enemy on the left, Boss on the right)*

## Weapons

There are 4 different types of weapons in the game and the player has access to them from the start. Each weapon can be equipped by using the number 1, 2, 3, 4 on their keyboard or by using the mouse wheel. Each weapon has its own statistics, these statistics are its range (how far it can shoot), its damage towards enemies, and its fire rate (how fast it can shoot/attack). The designs of the guns are made blocky and simplistic to match the same design as the game.

**Pistol**: The pistol (Figure 4.6) has the longest rage and can hit at target anywhere in the map but has low damage and a normal fire rate.

**Shotgun**: The shotgun (Figure 4.7) is a high damage weapon that can take away ¾ of an enemy’s health points in one shot. Its drawbacks are its low rage and slow fire rate.

**Automatic**: The automatic weapon (Figure 4.8) has a medium rage and really little damage. But its advantage is that is fire rate is fast and can shoot up to 10 times per second.

**Knife**: The knife (Figure 4.9) is not a gun but can exert devastating blows to enemies. One hit with the knife will kill and enemy, but its drawback is that its rage is quite low, requiring the enemy to be next to the player to land a hit.

A picture containing text

Description automatically generated A picture containing graphical user interface

Description automatically generated

*Figure 4.6(knife) Figure 4.7(shotgun)*

A picture containing text

Description automatically generated A picture containing engineering drawing

Description automatically generated

*Figure 4.8(automatic) Figure 4.9(knife)*

## Game Controls

**Movement:** For the user to move the player he must use the W, A, S, D keys on their keyboard to move around in the map**.**

W – Move forward

A – Moves to the left

S – Moves backwards

D - Moves to the right

**Jumping:** To jump in order to get over obstacles the ‘SPACE’ on the user’s keyboard needs to be pressed.

**Aiming:** To aim, the user needs to move around his keyboard.

**Shooting:** To shoot with any type of weapon the user must simply click the left button on his mouse. If the player has the automatic weapon equipped, he can hold the left mouse button down to make is weapon shoot multiple times.

## Pause Menu

When the player needs to take a break and pause the game, he can do that by pressing the ‘Escape’ key on his keyboard. This will stop the game that is in progress and show the pause menu (as show in Figure 4.10)

**Resume**: The Resume button when clicked will simply resume the game exactly where the player paused the game.

**Restart**: The Restart button when clicked will restart the game from the beginning. Taken back to full health and the enemies will be on level 1.

**Quit**: By pressing the quit button the game will quit and close the application.

A screenshot of a video game

Description automatically generated

*Figure 4.10(Pause Menu)*

## Map Design

The map design, alongside the enemies and the weapons was made to look simplistic and blocky. That allows the player to easily navigate along the map while also looking aesthetically pleasing.

Lighting: In each corner and edge of the map a spotlight has been placed to make the map fully visible to the player

Pillars: (Figure 4.11) Where used to block enemies and to take a tactical advantage point against them when walking around them

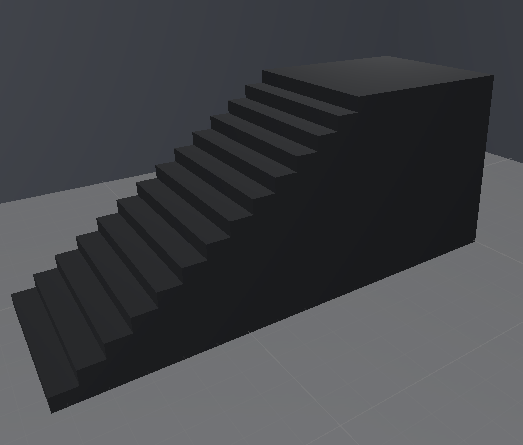
Stair Blocks: (Figure 4.12) Are an object mostly used to lure the enemies to one place and then escaping by jumping off the platform.

Bridges: Bridges (Figure 4.13) are the biggest object in the game, you can go over them and under them to get a tactical advantage.

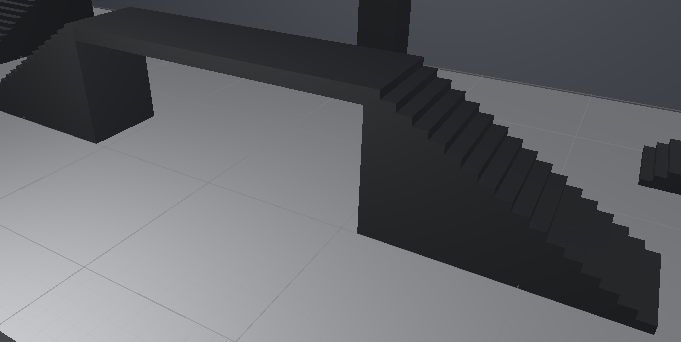
A picture containing shape

Description automatically generated

*Figure 4.11(Complete Map)*

A picture containing text, wall

Description automatically generated *Figure 4.12(Stair Blocks) Figure 4.11(Pillars)*



*Figure 4.13(Bridge)*

## Levels

**Number of Levels**: There are 10 levels in this game with each level being different to each other.

**Difficulty**: Starting from the first level the enemies will be slow and deal low damage. As you progress to the next levels enemies will increase, do more damage, and move faster. This increases the difficulty completing the level. The hardest level of all is the final level that also includes the ‘Boss’.

Diagram

Description automatically generated

*Figure 4.14(Final level with the ‘Boss’)*

## Sound

Sound has been included in many places in the game. Each weapon has its own unique sound, including a swift sound when changing weapons.

Jumping will also create a sound to produce a more immersive feel to the game.

## Game End states

The game has two possible outcomes. The one where you lose and the one where you win. If you lose you are greeted with the Game Over screen and if you beat all 10 levels, you are given the Victory screen.

**Game Over Screen**: (Figure 4.15) Comes up when you lose all your health points when the enemies damage you. By pressing the ‘SPACE’ bar on the keyboard will take you back to the main menu.

**Victory screen**: (Figure 4.16) Will come up once you have killed all enemies from all the levels. To exit the screen, you press the ‘SPACE’ bar and you are taken back to the main menu.

A picture containing diagram

Description automatically generated

*Figure 4.15 (Game Over Screen)*

A picture containing shape

Description automatically generated

*Figure 4.16(Victory Screen)*

# Implementation.

## Implementation Introduction

Having designed all the requirements for the project it was time to implement them to the project. The Implementation part of the project was the longest part due to the multiple increments that had to be done.

Since an incremental approach was used to develop the project, it was divided into smaller increments. This made it easier to focus on one thing at a time and if a requirement was to be altered to was easy to change.

Through the duration of the project challenges were faced but were also resolved using reason with the time that was remaining.

## Learning Unity

Learning unity was a tough experience at first and looked a bit intimidating. After a few videos and tutorials on the internet and using Unity’s own documentation it was easy to keep the ball rolling learning new stuff. Navigating through the scene manager was fun and almost game like experience. After learning all the shortcuts in Unity’s interface, it was time to take on the project.

## Main menu

The creation of the main menu UI was by far the easiest part of the project. By using Unity’s in-built UI tools (Figure 5.1), the menu was up and running withing the first week of implementation. Adding a button that started the game meant it was time to start working on the game.

Graphical user interface, application

Description automatically generated

*Figure 5.1(Unity’s in-built UI tools)*

## Player controls

Moving the player is a core part of the game. Meaning it should be the first thing to implement in the game.

Initially the player was going to move using the arrow keys on the keyboard, but that would make it difficult for the user to use the mouse. So, the requirements changed to use the W, A, S ,D keys instead. In result giving the player a more relaxed position to play the game on his desk. The left hand on the W, A, S ,D and SPACE bar keys and the right hand on the mouse.

## Enemies

After the player was free to move around in the map, creating enemies was the next step in the project. The goal was to make the enemies follow the player to anywhere he moved. At first, I coded a script that made the enemies follow the player to wherever he was standing but, there was a problem. If there was an object between the player and the enemy, they enemy would just crash onto the object and not go around it.

Therefore, the decision was made to use Unity’s built-in navigation Agent. This calculates the fastest possible route for the enemy towards the player by also avoiding objects in its path.

When they enemies got close to the player they will stop and attack they player while he is in range for them to attack.

Graphical user interface, application

Description automatically generated A picture containing sky, outdoor, light, day

Description automatically generated

*Figure 5.2 + Figure 5.3(Unity’s in-built Navigation Agent)*

## Weapons

Now that the player and the enemies were created it was time to make some weapons. Four different types of weapons were created with different characteristics. They all have different rage, damage and fire rates to make them unique to one other.

A single script was made to be used by each of the weapons, but all used different numbers for their variables in the class (Figure 5.4).

Text

Description automatically generated

*Figure 5.4(Variables used in the Shooting script used by the weapons)*

## User Interface

The user interface while in the game as mentioned in the design and build chapter 4, consisted of the player health bar, the current level, and the weapon equipped statistics.

Graphical user interface

Description automatically generatedAgain, using Unity’s UI tools, the level displayed at the top of the screen, the health bar at the bottom of the screen and the weapons statistics at the bottom left of the screen to be clearly visible to the player(Figure5.5) .

*Figure 5.4(User Interface)*

## Game design

With most of the functionalities of the game have been realized it was time to implement these functions together to create the game.

A script was made to load next levels after current level enemies were all dead. Healing the player to full health after each round to have better odds against they enemies was found to be too strong for the user. There were two options, either not regenerating the player to full health or making the next level enemies stronger. Decision was made to just not heal the player back to full health because it was the least time-consuming option of the two.

## Implementation Conclusion

During the implementation of the project many challenges were faced that have been resolved. Working on the challenges faced made me work harder to find better solutions thus increasing my knowledge in Unity and in C#.

After everything was put together the only thing left was to test the system functionalities (can be seen in chapter 6).

# Testing

## Testing Description

This chapter is for the formal testing of this project. Informal testing was being carried out while the project was being developed.

Unit testing has been carried out for the entire project to ensure that everything was working fine. With unit testing all the functionalities of the project were tested to see if the application was ready for deployment.

Tests were carried out on the ‘Main Menu’, the Game itself and on the ‘Pause Menu’.

A table was created to document what tests have passed and failed.

The table includes:

Test Number – The number of the test.

Test Description – A short description of the test to be carried out.

Expected outcome -The expected outcome of the test if it passed.

Result – The result of the test, this could be either a pass or a failure.

Issue – The issue with the test, this applies only to failed tests.

Solution – The solution to fix the reason the test that failed.

## Unit Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Number | Test Description | Expected Outcome | Result | Issue | Solution |
| Main Menu | | | | | |
| 1 | Open Application | Application loads | Passed |  |  |
| 2 | Quit Button | Application closes | Passed |  |  |
| 3 | How to Play button | Load ‘How to play’ page | Passed |  |  |
| 4 | Back button | Goes back to main menu page | Passed |  |  |
| 5 | Play Game button | Game loads on level 1 | Passed |  |  |
| Game | | | | | |
| 6 | Right click | Weapon that is equipped attacks | Passed |  |  |
| 7 | W key pressed | Player moves forward | Passed |  |  |
| 8 | A key pressed | Player moves to the left | Passed |  |  |
| 9 | S key pressed | Player moves backwards | Passed |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 10 | D key pressed | Player moves to the right | Passed |  |  |
| 11 | SPACE bar key is pressed | Player jumps in the air | Failed | Sometimes player would not jump | Fixed bug in code |
| 12 | ‘1’ key is pressed | Corresponding weapon is equipped | Passed |  |  |
| 13 | ‘2’ key is pressed | Corresponding weapon is equipped | Passed |  |  |
| 14 | ‘3’ key is pressed | Corresponding weapon is equipped | Passed |  |  |
| 15 | ‘4’ key is pressed | Corresponding weapon is equipped | Passed |  |  |
| 16 | Scroll wheel moved | Weapon equipped is changed | Passed |  |  |
| 17 | Player moves on platform | Player stays on platform and does not go through | Failed | Player fell through the platforms | Fixed platforms to be recognized |
| 18 | Enemies follow player | Enemies should follow player | Passed |  |  |
| 19 | Enemies find optimal path | Enemies should follow most optimal path to player | Passed |  |  |
| 20 | Enemy damage | Enemies take damage when shot | Passed |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 21 | Enemy health bar decreases | Enemy health bar decreases when shot | Passed |  |  |
| 22 | Enemy dies | Enemy dies when health goes equal or below 0 | Passed |  |  |
| 23 | Enemy attack | Enemy attacks player | Passed |  |  |
| 24 | Player Health bar | Player health bar decreases when attacked | Passed |  |  |
| 25 | Game Over screen | Game stops and shows Game over screen when player health is 0 | Passed |  |  |
| 26 | Next level loaded | After all enemies die never level should be loaded | Passed |  |  |
| 27 | Levels load enemies | When levels are loaded, they should load all enemies in that level | Passed |  |  |
| 28 | Victory screen | Victory screen is showed when player beats the final level | Passed |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 29 | Space bar key is pressed when death or victory screen is showed | Should return the player to main menu | Passed |  |  |
| 30 | ‘Escape’ key is pressed | Game should be pause and ‘pause menu’ should show | Passed |  |  |
| Pause Menu | | | | | |
| 31 | ‘Escape’ key is pressed again | Game should resume and pause menu disappears | Passed |  |  |
| 32 | Resume button | Game should resume and pause menu disappears | Passed |  |  |
| 33 | Restart button | Game starts from the beginning at level 1 with player at hull health points | Passed |  |  |
| 34 | Quit button | Application Closes | Passed |  |  |

# Evaluation

## Skills acquired.

Through this journey of developing this project I have acquired a few skills.

**Learning Unity Environment**: The biggest skill I have gained is without a doubt learning how to use unity to create a game. It is a useful tool that has a lot of functionality to help you build a project of your liking and a lot of flexibility to be creative.

**Improving my knowledge in C#:** I have had some knowledge using C# scripts in the past but undertaking a project of this magnitude has made me push my boundaries to learn more about the language and its capabilities.

**Implementing Methodologies:** Implementing methodologies in projects was something new for me as I have never really had a method on how I completed a project. By researching different methodologies and use them in my project made me realize the drastic impact they can have in the success of the project

## Requirements

Requirements met in Chapter 3(Requirements) have mostly been met to a high standard throughout the project but there should always be some room for improvement.

## Future Additions

Future additions to this project could involve adding more weapons to the game. Maybe adding new maps with different type scenery.

A more exciting addition could be new game modes including multiplayer that could be played with friends. But obviously these additions could not be implemented in this project due to time limitations.

## Conclusion

In conclusion this project was a success since it has met most of the initial requirements and has left room for more development in the future if needed. Gaining new skills in the duration of this project was the best part but also the result was also something to be proud of.

# References

[1] “Waterfall Project Management Methodology,” [Online]. Available: <https://activecollab.com/blog/project-management/waterfall-project-management-methodology> [Accessed 15 April 2021]

[2] [Online]. Available: https://learn.g2.com/waterfall-methodology [Accessed 15 April 2021]

[3] “What is agile methodology?” [Online]. Available: https://www.infoworld.com/article/3237508/what-is-agile-methodology-modern-software-development-explained.html [Accessed 15 April 2021]

[4] [Online]. Available: https://hackernoon.com/a-case-study-type-insight-into-agile-methodologies-for-software-development-cd5932c6 [Accessed 15 April 2021]

[5] “Incremental Model,” [Online]. Available: https://www.javatpoint.com/software-engineering-incremental-model [Accessed 15 April 2021

[6] [Online]. Available: http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 15 April 2021]

[7] “What is Unity 3D & What is it Used For?” [Online]. Available: https://conceptartempire.com/what-is-unity/ [Accessed 15 April 2021]

[8] [Online]. Available: https://unity3d.com/beta/2020.1b [Accessed 15 April 2021]

[9] “What is Unreal Engine?” [Online]. Available: https://conceptartempire.com/what-is-unreal-engine// [Accessed 15 April 2021]

[10] [Online]. Available: https://www.unrealengine.com/marketplace/en-US/product/21cab2d84a794b7bab359eb1ba5d3e74 [Accessed 15 April 2021]

[11] “What’s the Difference Between C# and C++? “[Online]. Available: https://www.guru99.com/cpp-vs-c-sharp.html [Accessed 15 April 2021]

[12] “MoSCoW Method,“ [Online]. Available: https://www.projectsmart.co.uk/moscow-method.php [Accessed 15 April 2021]