HAND GESTURE CONTROL FIRST-PERSON SHOOTER GAME

GOH CHIANG CHENG

UNIVERSITI TEKNOLOGI MALAYSIA

PSZ 19:16 (Pind. 1/13)

UNIVERSITI TEKNOLOGI MALAYSIA

DECLARATION OF THESIS / UNDERGRADUATE PROJECT REPORT AND COPYRIGHT

Author's full name	: GOH CHIANG CHENG
Date of Birth	: 6/1/1997
Title	: HAND GESTURE CONTROL FIRST-PERSON SHOOTER GAME
Academic Session	: 2020/2021

I declare that this thesis is classified as:

☐	**CONFIDENTIAL**	(Contains confidential information under the Official Secret Act 1972)*
☐	**RESTRICTED**	(Contains restricted information as specified by the organization where research was done)*
✓	**OPEN ACCESS**	I agree that my thesis to be published as online open access (full text)

1. I acknowledged that Universiti Teknologi Malaysia reserves the right as follows:
2. The thesis is the property of Universiti Teknologi Malaysia
3. The Library of Universiti Teknologi Malaysia has the right to make copies for the purpose of research only.
4. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

SIGNATURE OF STUDENT	**SIGNATURE OF SUPERVISOR**
A17CS0048	DR. JUMAIL BIN TALIBA
MATRIX NUMBER	**NAME OF SUPERVISOR**
Date:	Date:

NOTES : If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentiality or restriction

"I hereby declare that we have read this thesis and in my
opinion this thesis is suffcient in term of scope and quality for the
award of the degree of Bachelor of Computer Science (Graphics and Multimedia
Software)"

Signature	:	______________________________
Name of Supervisor	:	JUMAIL BIN TALIBA
Date	:	

HAND GESTURE CONTROL FIRST-PERSON SHOOTER GAME

GOH CHIANG CHENG

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Bachelor of Computer Science (Graphics and Multimedia Software)

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia

AUGUST 2021

DECLARATION

I declare that this thesis entitled *"Hand Gesture Control First-Person Shooter Game"* is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature	:	…………………………………………….
Name	:	GOH CHIANG CHENG
Date	:	

DEDICATION

This thesis is dedicated to all lecturers in Universiti Teknologi Malaysia (UTM) who did share their knowledge to me during my study in the university.

ACKNOWLEDGEMENT

I would like to express my appreciation to all the lecturers in School of Computing that did provides useful knowledge and guidance to me for completing my study in Universiti Teknologi Malaysia (UTM). I would also like to thank my supervisor, Dr. Jumail Bin Taliba for his encouragement, patience, and support. His guidance helped me a lot in deciding the title for this thesis and technique that is related to it.

In addition, I would like to express my gratitude to the PSM committee of School of Computing for providing the guideline and material that is needed for this thesis. The content provided by them are well-organized and easy to understand.

Finally, my thanks will be given to my friends and family for their continuous support and caring. Their support has motivated me to work harder on completing this thesis.

ABSTRACT

The purpose of this study is to develop a First-Person Shooter (FPS) game which can be control through hand gesture without the use of hardware such as Kinect sensor. The concept of the project is to implement the knowledge of image processing to detect the hand gesture that has been captured by the webcam. Hand gesture is a more natural communication medium that the player can send instruction to the computer in compare with the use of traditional input device such as mouse and keyboard. There will be two subparts during the development of the proposed system which is the FPS game development and hand gesture detection development. These two parts will be evaluated separately, and the final evaluation will be done again after the integration of the system. This project is expected to prove that the use of webcam as the hand gesture detection device does have the potential to decrease the cost that needed for the hand gesture sensor while provides the features which is similar like it.

ABSTRAK

Kajian ini dilakukan bertujuan menghasilkan satu permainan video berjenis penembak diri-pertama yang boleh dikawal melalui isyarat tangan tanpa menggunakan perkakasan seperti sensor kinect. Konsep projek ini adalah menggunakan pengetahuan pemprosesan imej untuk mendapatkan informasi isyarat tangan yang ditangkap oleh kamera web. Isyarat tangan adalah jenis medium komunikasi yang lebih mudah dan selesa yang boleh dilakukan oleh pemain untuk menberi arahan kepada komputer berbanding dengan medium komunikasi biasa seperti penggunaan tetikus dan papan kekunci. Sistem yang dicadangkan terdapat dua bahagian iaitu bahagian penghasilan permainan video berjenis penembak diri-pertama dan bahagian pengesan isyarat tangan. Kedua-dua bahagian akan dinilai secara berasingan dan penilaian yang akhir akan dibuat selepas bahagian tersebut digabungkan. Projek ini dijangka dapat membuktikan bahawa kegunaan kamera web sebagai pengesan isyarat tangan mempunyai potensi untuk mengurangkan kos untuk sensor isyarat tangan dengan memperoleh fungsi yang serupa.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FPS	-	First-Person Shooter
PC	-	Personal Computer
NPC	-	Non-Player Character
3D	-	Three Dimensions
API	-	Application Programming Interface
OpenCV	-	Open Source Computer Vision Library
IDE	-	Integrated Development Environment
OS	-	Operating System
RAM	-	Random Access Memory
UI	-	User Interface
ROI	-	Region Of Interest

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

According to Kuikkaniemi et al. (2010, April), first-person shooter (FPS) games is a well-known video game category which focus on the first-person perspective and provides a gaming platform of firearm-based combat. The first-person perspective is set on an avatar's point of view which allows the player to experience the game through the view of the avatar to provide a better immersive feel that the player can observe and interact with the environment around the avatar (Denisova and Cairns, 2015, April). Normally, the FPS games will let the player able to see an arm (or pair of arms) which is either empty-handed or holding any type of weapon such as pistol or knife (Grimshaw, Charlton and Jagger, 2011). This weapon will let the player able to interact with the target in the games which can be either programmed character or player. One of the simplest interactions is to aim and shoot the target until the target is taken down.

Hand gestures is a simple communication method for the user to interact with the computer and no advance training is needed for the user to understand the way to communicate with the computer (Hamza, Anand, Shvhare and Gaurav, 2017). By using hand gestures interaction, the user will only need to know the pre-defined hand shape and movement that match the instruction set up by the programmer to execute certain command to the computer. According to Dardas, Silva and El Saddik (2014), The advantages of hand gestures control in video game is that the user can interact with the computer without physical contact and make the interaction process become more natural, immersive and comfortable.

1.2 Problem Background

The improvement of technology in computer science makes the customer in gaming industry wants to play the computer games that involved body movements which is close to the actual ways the people will do in real life (Roccetti, Marfia and Semeraro, 2012). Some user feels bored in playing games through traditional interface such as keyboard and mouse. Hand gesture control can let the user play the video game with some predefined hand pattern and movement which is more realistic and entertaining to play. This is because hand gestures are considered as the most natural and flexible communication media to transfer the command to the computerized devices in compare with the other interface such as keyboard or mouse (Chen, Lin and Yang, 2010, June). According to Roccetti et al. (2012), although the price of the equipment that act as the sensor of the hand gestures such as Leap Motion Controller and XBOX360 Kinect Sensor has gradually decreased, it is still more worthy if there is a solution to make the simple webcam act as the device to capture the hand gesture input. Although webcam cannot provide the depth information of the environment, but the use of image processing can still detect the hand gesture with the information provided by the webcam in real-time manner (Hamza et al., 2017).

1.3 Project Aim

The aim of this project is to develop a first-person shooter game which can be played through hand gesture with the input from webcam.

1.4 Project Objectives

The objectives of the project are:

(a) To analyse the requirements of first-person shooter games controlled using hand gestures.

(b) To design a personal computer (PC) game controlled by hand gestures.

(c) To implement the image processing algorithm through input from webcam.

(d) To evaluate the potential of webcam that can replace the use of other hand gesture sensor.

1.5 Project Scope

The scopes of the project are:

(a) The game produced will be in single player and personal computer based which will fully controlled by hand gesture throughout the whole gaming process.

(b) The hand gestures input will only focus on hand shape and movement only.

(c) The character in the game would not be able to move and they need take down all the enemies in the scene to go to the next stage. A shield command will be provided to block enemies attack.

(d) The game will in stage form which let the player can start at any stage that they clear before.

(e) The image input will be gathered only through webcam without any others image input devices.

1.6 Project Importance

The project is important to use image processing algorithm to identify the hand gesture that has been captured by the webcam to interact with the FPS game. The hand gesture should be more relevant that can let the player feel comfortable and enjoy the gaming process. The main idea of the project is to use webcam approached to offer the functionality of Kinect sensor in order to decrease the cost needed for game which is controlled using hand gesture.

1.7 Report Organization

Chapter 1 is the introduction of the project and it has been described in the section above. The next chapter is Chapter 2 which describes the literature review. Chapter 3 will explain about the project methodology and the proposed system will be explain in Chapter 4. Chapter 5 will make a conclusion about the project.

Chapter 2 is the literature review of the project. This chapter describes about first-person shooter (FPS) game, hand gestures and the comparison with existed system. The technology used will be review at this chapter.

Chapter 3 explain about the project development methodology. This is the project guideline that use to define the procedure on implementing and developing the project to achieve the aim of the project.

Chapter 4 will state the requirement analysis and the design of the project. The design will be described in this section includes project design and interface design.

Chapter 5 concludes the project by describing the expected achievements of the project objectives, constraints and state the planning for PSM 2.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter describes the terms of First-Person Shooter (FPS) game and explain about hand gesture which includes the importance of hand gesture in video game. Lastly, some previous works which is similar like the project will be reviewe d and compare with the proposed system.

2.2 First-Person Shooter game

The subsections will describe the definition, characteristic and requirement of FPS game.

2.2.1 Definition of FPS game

According to Hitchens (2011), FPS is a genre of video game which use first-person viewpoint as scene display that is heavily focus on the combat between player and player or player and non-player character (NPC) which is mainly using firearms. The use of ranged weapon to start the combat between the player and the opponent will create a more rapidly changing environment which required the player to react fast and accurately so that they can take down the opponent as fast as possible to prevent taking too much damage from the opponent (Colzato, Van Leeuwen, Van Den Wildenberg, Hommel, 2010). FPS game does provide a platform for those people who likes firearm to be able to use it in a virtual environment which fulfil their hype on using it.

2.2.2 Characteristic of FPS game

FPS games are normally considered as a violent video game due to the involvement of guns and the action of taking down enemy through firearms (Weber et al., 2009). The first main genre of FPS games is single player mode which provides a storyline for the player to explore and immerse themselves into the experiences of the avatar they are controlling. According to Cardamone, Yannakakis, Togelius and Lanzi (2011, April), this kind of FPS game will have a limited playing time around 10 hours depends on the budget and the storyline the developer wishes to present to the player. The player will only face NPC which has applied the computational intelligence techniques as the basic of its behaviour. The second main genre is multiplayer mode, the player needs to compete against each other to take down the human controlled avatar that are not same team as them or fighting against the NPC together to achieve the mission that is being given by the system (Zhou, Miller and Bassilious, 2008, May).

2.2.3 Requirement of FPS game

According to Therrien (2015), the FPS game is always bind to the terms, “perspective”, “3D” and “simulation”. The mainframe game Maze War (1974) which is considered as the earliest FPS genre game is a good example that explain how a game can be defined as the FPS game. In term of perspective and simulation, the player should have the ability to perform task with the viewpoint of the avatar inside the FPS game depends on the command given to the avatar as in Maze War. For instance, the scene of Maze War will change when the moving instruction has been applied or the player give the command to fire a shot, a bullet will also be fired. The ”3-D” term is the visual representation form of the environment of the FPS game. The scene should be able to provides some interaction that is close to the understanding of player on the real 3D world such as the wall will block the movement and view of sight of the player as in Maze War (Grimshaw et al., 2011) which has been shown in Figure 2.1.

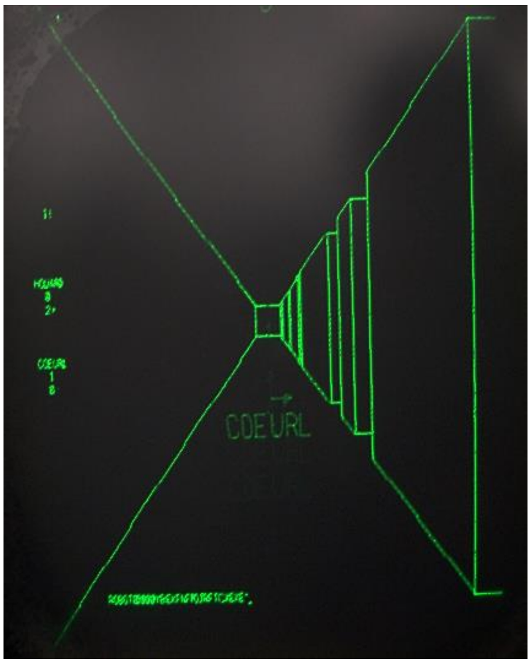


Figure 2.1: Maze War (Grimshaw et al., 2011)

Besides the fundamental requirement of FPS game, the game content which includes maps and levels into the FPS game is crucial to optimize the player experience contemporary (Cardamone et al., 2011, April). The FPS game that the player only can go through the same map and shoot the same types of bot will not be able to provides the joy that the player needed from the game. Zhou et al. (2008, May) did state that the internet delay must be considered for the multiplayer FPS game since this will affect heavily on the result of the game.

2.3 Hand Gesture

The subsections will describe the definition of hand gesture, hand gesture recognition and the importance of hand gesture in video game industry.

2.3.1 Definition of Hand Gesture

According to Chen et al. (2014), gesture is a result of a human behavior and a way that human use to express their emotion and feeling. There is two main type of gesture which is hand gesture and body gesture that both can be use on sending useful message to others. Hand gesture is a type of compressed information which can only be understood by those who knows how to decode the gesture presented (Ma'asum, Sulaiman, Saparon, 2015). The change in the shape and movement of the finger will define the hand gesture but the same hand gesture may have different meaning in terms of the understanding and knowledge of the sender and the condition of the situation.

2.3.2 Hand gesture recognition

There is three main process that will gone through to detect and recognize the hand gesture presented by the user which are camera module, detection module and feature extraction module (Ma'asum et al., 2015) while the flow of the steps has shown in Figure 2.2. In camera module, the image contains the hand gesture will be capture by the webcam and converted into digital information which will be use on detection module to retrieve the data of the threshold images. The system will identify the region in the image that needed to be focus on which is the hand region so that the data in that specific region will be emphasized and take as the primary data that will be use during the Feature Extraction Module (Chen et al., 2014). During the Feature Extraction Module, the feature of the hand region will be extracted and use as the description of the hand gesture. After these three modules, the feature information will be given to the program to compare with the data inside of the feature database to calculate the similarity between each dataset and identify which hand gesture has been used in the image.

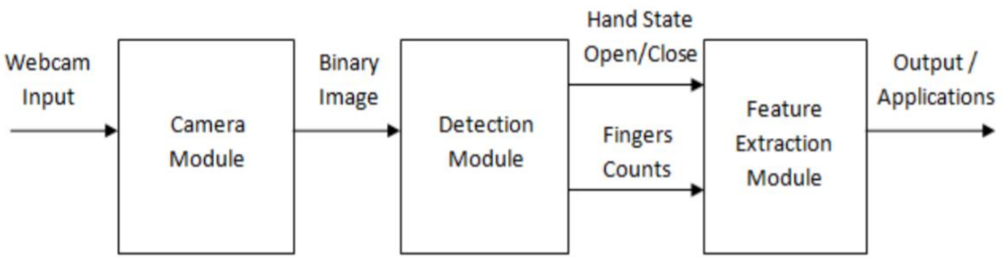


Figure 2.2: The framework of hand gesture detection (Ma'asum, Sulaiman, Saparon, 2015)

2.3.3 Importance of hand gesture control in video game industry

The use of gesture-based interfaces which is closer to the real-life interaction has become one of the primary factors that can determine the success of a video games (Roccetti et al., 2012). The user starts to find the user interface that is less reliable on traditional interfaces like joystick, keyboard or mouse but wants to play the video games with the ways that is far more natural and closer to the action in the reality. For instance, they wish to grab something in the video game with grabbing hand but not left click the mouse or looks around by moving their head but not moving the mouse. According to Chen et al. (2010, June) although a glove-based approach did propose in some video games, but the user still not satisfy with the needs on wearing the glove since this input interface required the player to have addition equipment that they need to wear on.

2.4 Existed System

In this section, 2 existed system which is webcam-based hand gesture control in Skyfall and G-shooting Gesture will be introduced.

2.4.1 Webcam-based hand gesture control in Skyfall

Skyfall is a kind of web-based game that the player can play it using the browser. The ways to play the game is to control the paddle left and right to collide with the ball that is falling down. White ball and green ball will give the player 10 points while red ball will decrease 10 points from the player. The standard Skyfall interface is using mouse to control the movement of the paddle, but the developer did add in the hand gesture control that get the image input through webcam to detect the hand movement of the player to control the paddle. The sample image has shown in Figure 2.3.

This application use TensorFlow object detection application programming interface (API) as the machine learning library and trained with the models of the hand that the application needs to detect and recognize. The detected hand gesture will be sent by the API to the game interface as the input to control the paddle without the needs on using mouse.

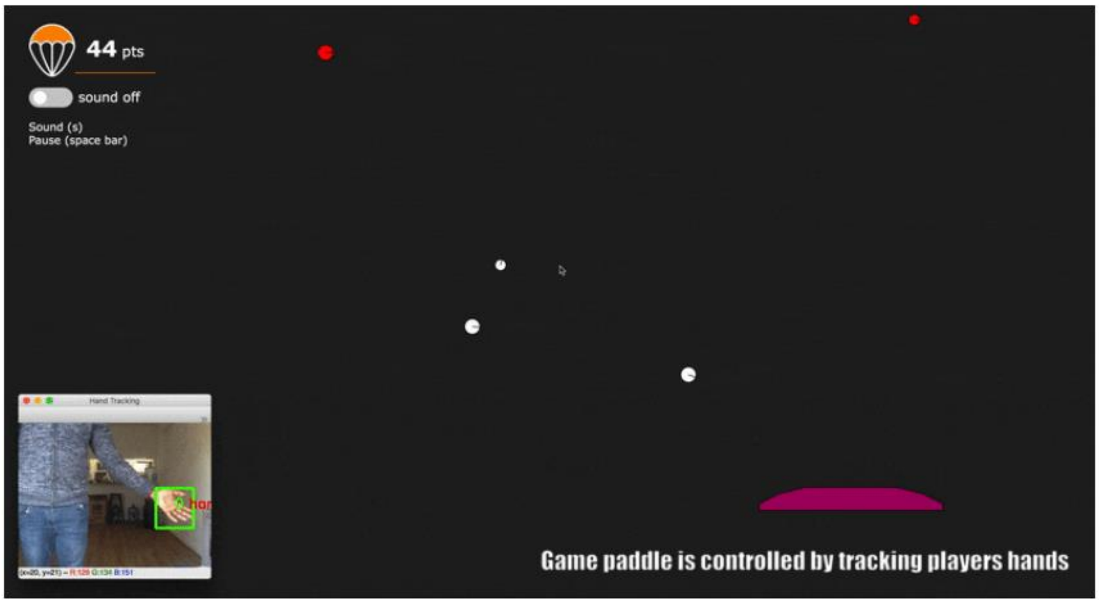


Figure 2.3: Hand gesture control in Skyfall

2.4.2 G-Shooting Gesture

G-Shooting Gesture is a mobile game which is able to be download in Google Play Store. It is a kind of FPS game that the player can control the gun in the game through the camera of their mobile phone by using hand gesture. The hand gesture needed for the controlling is using their right hand's index finger. The cursor in the application will follow the movement of the index finger and start shooting when the index finger has stop moving. Figure 2.4 is the image which show the sample gameplay of G-shooting Gesture.

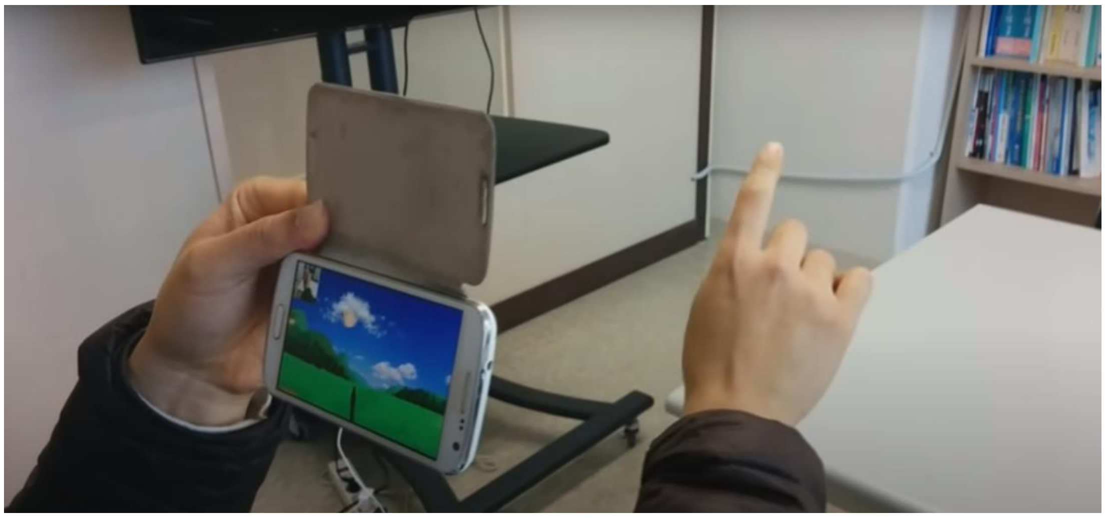


Figure 2.4: Sample gameplay of G-shooting Gesture

2.5 Comparison between existing systems

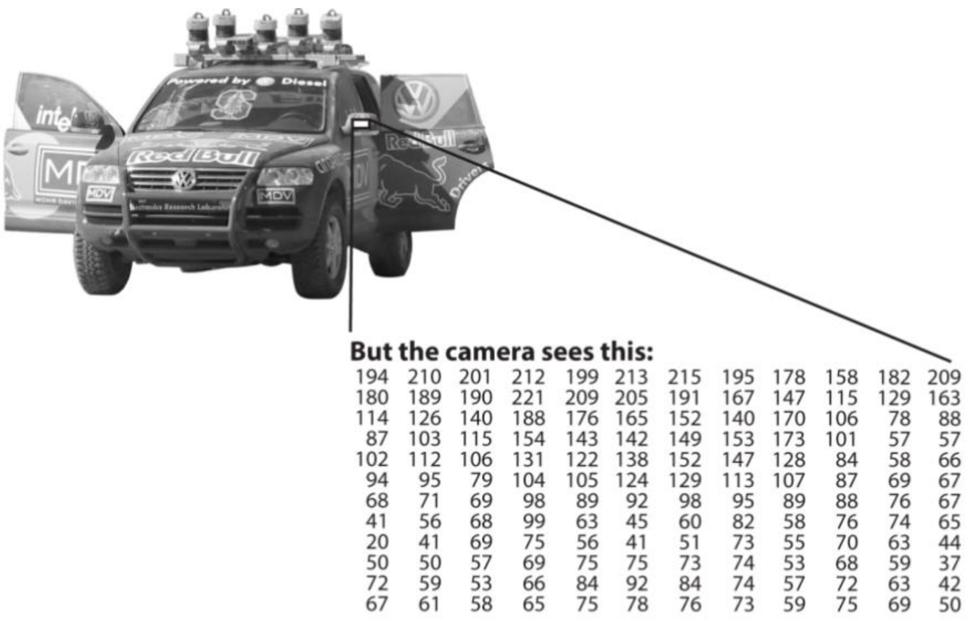
The 2 existing system that stated in Chapter 2.4 does have the similar concept with the proposed system in this project which is detect the hand gesture of the user without the assistance from motion sensor hardware. The image of the hand gesture will capture through normal camera and recognized by the system through image recognition. The differences between the two existing system and the proposed system are the platform of the game, the proposed system will be a personal computer (PC) game.

2.6 Literature Review of Technology Used

This section will define the technology and platform that will be uses to develop the proposed system which is OpenCV library and Unity 3D.

2.6.1 Open Source Computer Vision Library (OpenCV)

OpenCV is an open source library mainly written in C and C++ which has been further developed into integrate with others programming interface such as Python, Java and Matlab. According to Kaehler and Bradski (2016), the purpose of OpenCV is to enhance the usage of computer vision and artificial intelligence which can be processed in real-time application. OpenCV is used to fulfill the aim of computer vision by understanding the data contains in an image and convert it into the knowledge on making some sort of decision. Figure 2.5 is an example of the needs of computer vision since the computer will not know that the image is a car mirror but just a matrix contains values. OpenCV will be able to determine that the image input is a car mirror through image processing or even machine learning.

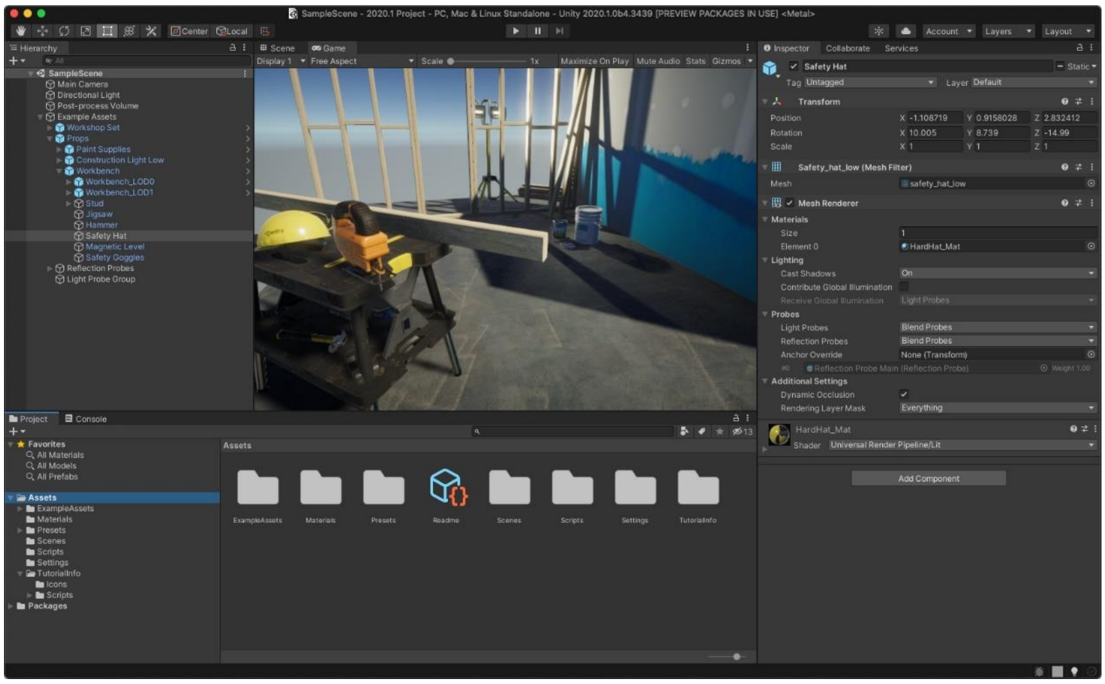

But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Figure 2.5: How computer understand about the image (Kaehler and Bradski, 2016)

2.6.2 Unity3D

Unity3D is a game engine that is being developed by David Helgason, Joachim Ante and Nicholas Francis which provides an integrated development environment (IDE) for developing video game (Haas, 2014). Unity3D is an open source IDE which provides the opportunities for independent developer to be able to produce interaction media without the needs on spending money on buying the license. Figure 2.6 is the sample interface of Unity3D and there are 2 core components that makes Unity3D a successful game engine which is the assets pipeline and multiplatform compatibility.



Safety Hat — Static
Tag Untagged — Layer Default

Transform
Position X -1.108719 Y 0.9158028 Z 2.832412
Rotation X 10.005 Y 8.739 Z -14.99
Scale X 1 Y 1 Z 1

Safety_hat_low (Mesh Filter)
Mesh: safety_hat_low

Mesh Renderer
Materials
Size: 1
Element 0: HardHat_Mat
Lighting
Cast Shadows: On
Contribute Global Illumination
Receive Global Illumination: Light Probes
Probes
Light Probes: Blend Probes
Reflection Probes: Blend Probes
Anchor Override: None (Transform)
#0 Reflection Probe Main (Reflection Probe) Weight 1.00
Additional Settings
Dynamic Occlusion: ✓
Rendering Layer Mask: Everything

HardHat_Mat
Shader: Universal Render Pipeline/Lit

Figure 2.6: Sample interface of Unity3D

The asset pipeline in Unity3D provides the possibility for the developer on taking the assets that has been created by others developer that shared on the Unity Asset Store to integrate with their project. These assets can be 3D model or script that the owner choose to publish in the Unity Asset Store as shown in Figure 2.7 with price or sometimes it will be free. This provides a platform for the developer to share their product, learn from others or even decrease the development of their video game since they may not need to produce those 3D model or script themselves and focus more on the other content of the games. There is also some asset is use for the tutorial so that the new developer can be familiar with Unity3D.

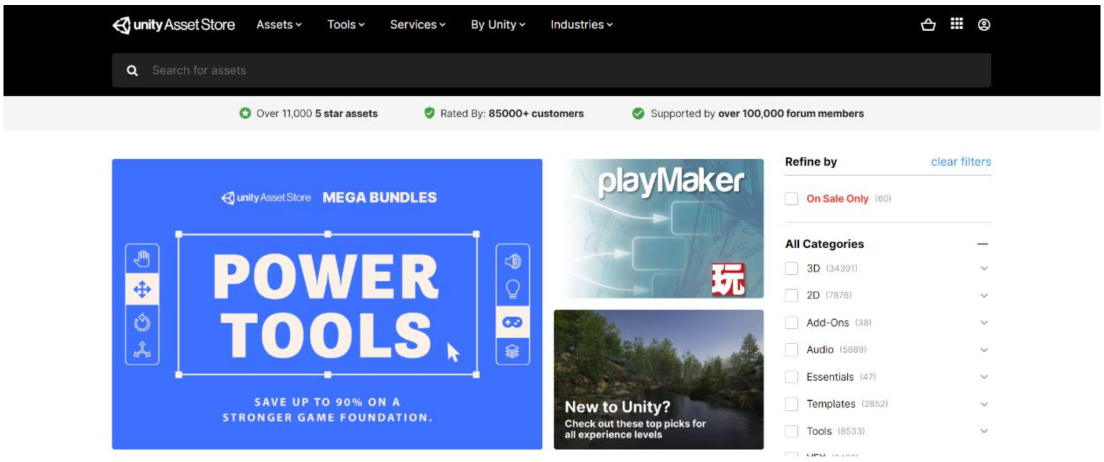
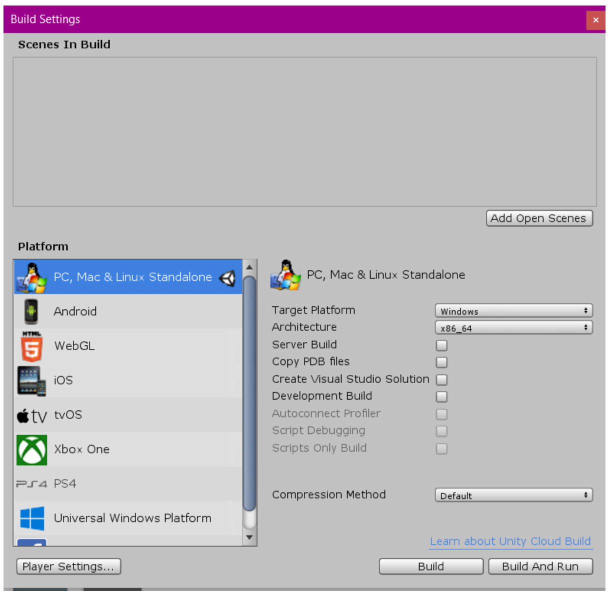


Figure 2.7: Unity Asset Store

Haas (2014) did states that Unity has become the favourite of the game developer is because Unity3D did provides the functionality on publishing the video game in different platform by using the same code and asset. These platforms included mostly all the current popular platform such as iOS, Android, Windows, Mac OS X and Linux. Figure 2.8 has displayed the multiplatform available in Unity3D which can decrease the needs of reprogramming the project so that the application can be run in different platform.



PC, Mac & Linux Standalone

Target Platform: Windows
Architecture: x86_64
Server Build
Copy PDB files
Create Visual Studio Solution
Development Build
Autoconnect Profiler
Script Debugging
Scripts Only Build
Compression Method: Default

Figure 2.8: Multiplatform available in Unity3D

2.7 Chapter Summary

Subchapter 2.2 had provided an explanation about the terms FPS game and what is the characteristic and requirement of an FPS game. FPS game is a genre of games that has many fans and is well-known among the gamers.

Subchapter 2.3 discuss about the definition of hand gesture and the ways to recognize hand gesture. The importance of hand gesture control in video game will also state in this subsection. 3 main components in recognizing hand gesture are camera module, detection module and feature extraction module. Hand gesture control is important in video game because of its real-life interaction which provides a more natural ways to interact with the video games.

Subchapter 2.4 list two existed system which is similar to the proposed project. One of the existed systems is webcam-based hand gesture control in Skyfall and the others is G-Shooting Gesture.

In Subchapter 2.5, the comparison between the existed system and proposed system has been done. The similarity of the system is that both using webcam as the input of the hand gesture, but the differences is the platform of the video games.

Subchapter 2.6 is the literature review of the technology used which is OpenCV and Unity3D. OpenCV will use on recognizing the hand gesture while Unity3D is the platform that use to create the FPS game.

CHAPTER 3

SYSTEM DEVELOPMENT METHODOLOGY

3.1 Introduction

This chapter will describe the system development methodology that has been used on developing the system which is agile development methodology.

3.2 Methodology Choice and Justification

The methodology used in this project is agile development methodology. The agile development methodology has decreased the potential risk to the project which includes change in requirement and adding new functionality. Agile method has successfully improved the quality of the system due to the small iteration of testing and requirement checking during system development. The FPS game and hand gesture detection has been tested a lot of times and many new functions has been added to enhance the performance of the system. Figure 3.1 shows the flow of the Project Methodology.

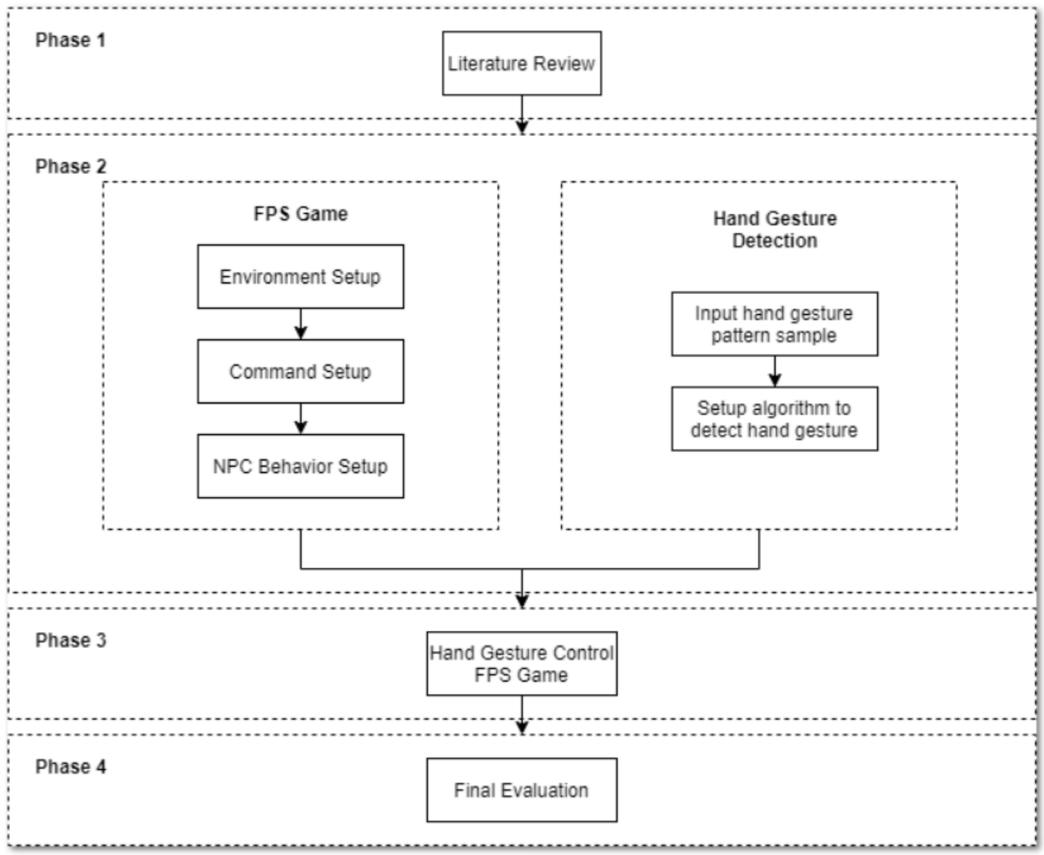


Figure 3.1: Project methodology

3.3 Phases of the Chosen Methodology

The 5 main phases in each iteration of the agile methodology are determine requirement, system design, implementation, testing and make improvement that has been shown in Figure 3.2. These 5 phases have been gone through in each iteration and the iteration does not change if the result does not fulfil the requirement.

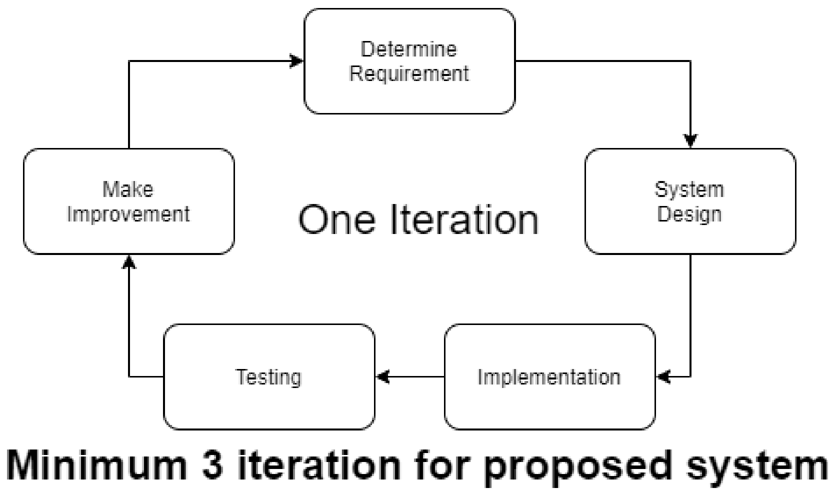


Figure 3.2: Phases of an iteration in agile methodology

The first phase is determining the requirement of the system which means that the functionality and the expected performance that the current iteration of system needs to fulfil have been identify. The system has been designed according to the requirement that did define at previous phase. Due to the use of agile methodology, the requirement at this spot has become the minor achievement of current iteration and increased after the next iteration. The system has been developed at implementation phase through coding or user interface creation which involve the actual use of computer to implement the algorithm.

The user has tested the iteration of the system to determine the performance of current system. The system has completed the improvement before going to the next iteration with the same cycles according to the user feedback. The proposed system does have two subpart which is the hand gesture detection section and FPS game section. Each section has gone through the iteration stated above and the combination of the subsection were going through again the same iteration.

3.4 Technology Used Description

There are two main components in this project which is OpenCV and Unity3D. Figure 3.3 is the concept image of the technology which has been used in this project. OpenCV is the open vision library use to detect and recognize the hand gesture that will be input from the webcam. The flow of the process is shown on Figure 3.4. The first step is capturing the image through the webcam and then the skin color detection has been done by using OpenCV. The skin color detection will identify the human body parts and further process will be done to remove the human body parts which is not the hand. Later on, the hand gesture will be filter into a grayscale image to make the hand gesture recognition system can be done more accurately and efficiently.

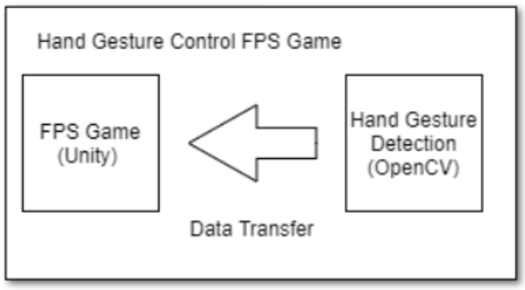

Hand Gesture Control FPS Game
FPS Game (Unity)
Hand Gesture Detection (OpenCV)
Data Transfer

Figure 3.3: Concept image of the technology use in the project

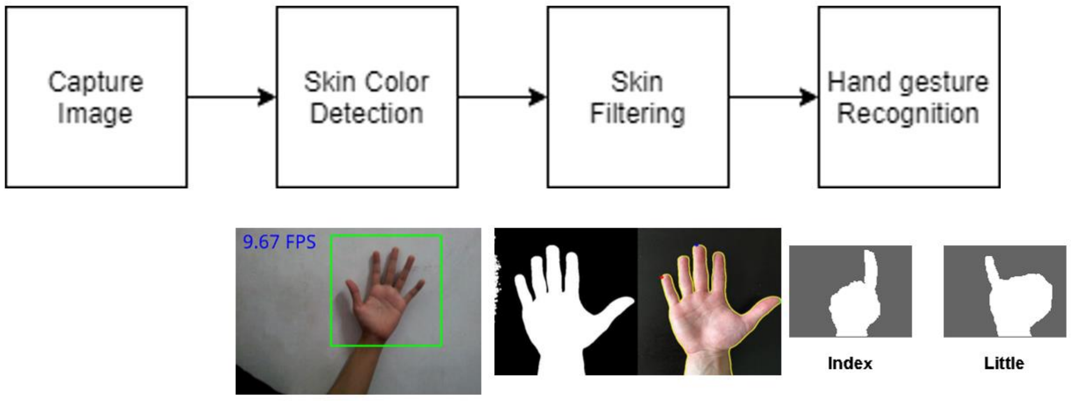


Figure 3.4: Flow of hand gesture detection and recognition in OpenCV

For the creation of FPS using Unity3D, a lot of assets have been created and insert into the scene to make the stage of the FPS game. The Unity3D does compile the application using its project build function. Microsoft Visual Studio has been used as the platform to modify the C# script in Unity3D. The input was gathered from OpenCV and the avatar in the game did act according to the input retrieve from OpenCV.

3.5 System Requirement Analysis

There is two main requirement that have been analyzed which is the hardware requirement and software requirement. Unity3D is the main parts of the development of FPS game, so the Operating System (OS) used did support Unity3D with version 2019.1. Table 3.1 list out the hardware requirement while Table 3.2 is the software requirement for the project.

Table 3-1: Hardware requirement

Component	Specification
OS	Windows 7 or above macOS 10.12 or above
RAM	4 GB DDR3 RAM
Graphic Card	Graphics card with DX10 (shader model 4.00) capabilities

Table 3-2 Software requirement

Software	Specification
Game Engine	Unity 2019.1
C# Script Editor	Microsoft Visual Studio 2017

3.6 Chapter Summary

Subchapter 3.2 has state the methodology choice for this project. Agile methodology has been selected for this project because the FPS game performance and the hand gesture detection can be evaluated separately.

Subchapter 3.3 show the phases for the chosen methodology. There are 5 phases for each iteration which is determine requirement, system design, implementation, testing and make improvement. The project will only go to the next iteration after the expectation and requirement for current iteration has been fulfilled.

Subchapter 3.4 describe the technology used in this project which is OpenCV and Unity3D. OpenCV is use on hand gesture detection while Unity3D is use on constructing the whole FPS game.

Subchapter 3.5 state the system requirement for the development of the project. Hardware requirement is determined by the compatibility with the Unity3D and OpenCV version.

CHAPTER 4

REQUIREMENT ANALYSIS AND DESIGN

4.1 Introduction

This chapter will provide the details about the function and addon used during the development of the project. The system development will be separated into 2 parts, which is the FPS Game part and Image Processing.

4.2 Requirement Analysis

The functional requirement and non-functional requirement that the completed system has been fulfilled has been listed at below.

4.2.1 Functional Requirement

(a) The completed system is able to detect hand gesture through webcam.

(b) The completed system is able to provides an FPS genre game for the player to play.

(c) The avatar in the FPS game does react correctly to the predefined hand gesture of the player.

(d) The player is able to shoot down NPC through hand gesture.

(e) The completed system does provide a tutorial for the player.

(f) The completed system does pinpoint the NPC that the player is currently aiming.

4.2.2 Non-Functional Requirement

(a) The completed system is able to recognize more than 2 types of hand gesture.

(b) The completed system is able to response to user's hand gesture in less than 2 second.

(c) The completed system does provide 3 stages for the player.

(d) The completed system is able to provide leaderboard for the user to determine the top scoring of the game.

4.3 Project Design

The whole system has been constructed under the game engine, Unity3D. All of the contents in the FPS game will be build and design using Assets in Unity and the behavior will be programmed using C# programming language. The image processing features will be implemented in Unity3D using a well setup wrapper, OpenCV+ Unity while the login features and leaderboard will be provided with the use of server platform, Microsoft Azure Playfab. The relationship of these components has been shown on Figure 4.1.

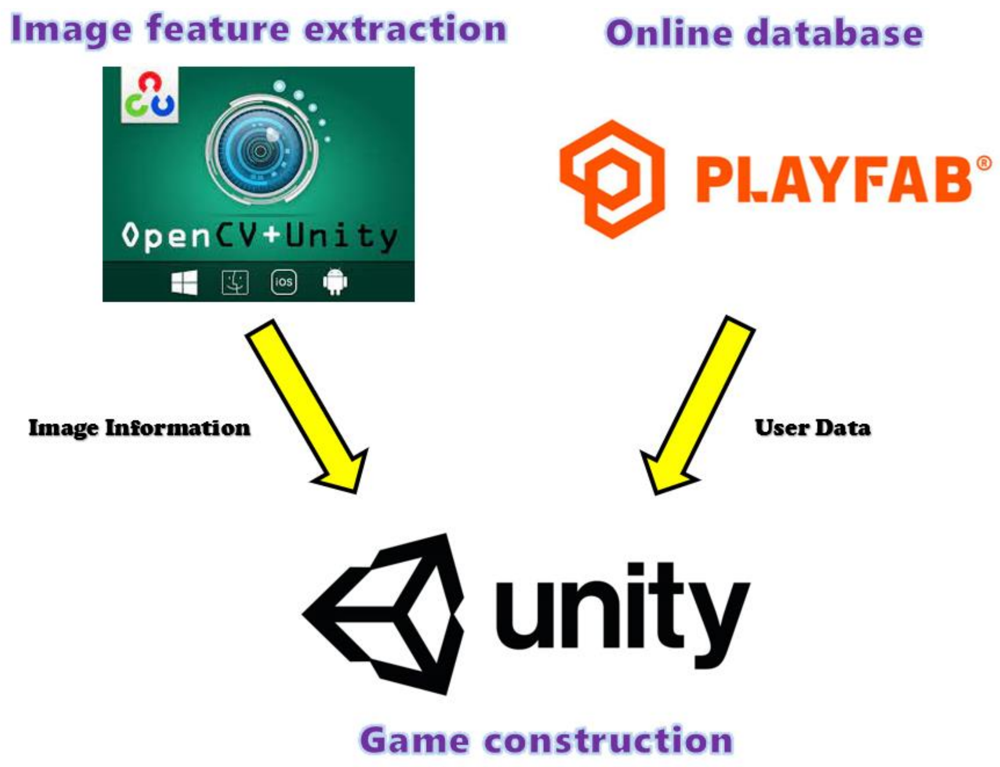


Figure 4.1: Components in Unity3D

4.3.1 Image Detection Algorithm

The image detection process will be done using the library provided in OpenCV+ Unity which has the similar function as in OpenCV with minor change in input and output type.

The first process of the image detection is identifying the webcam that will use as the input of the image. Next the algorithm will setup an area on the position of the webcam as the region of interest (ROI). Only the data in the ROI will extracted and send to further processing.

The first step for the image processing is convert the ROI image into grayscale image. Then the grayscale image will be blurred using Gaussian blur. These 2 steps are important since next step is changing the image into pure white and black image using the threshold function in OpenCV. The pixel value which is darker than the threshold value will become black color while the pixel with higher value will become white color. The sample result after this process has shown in Figure 4.2.

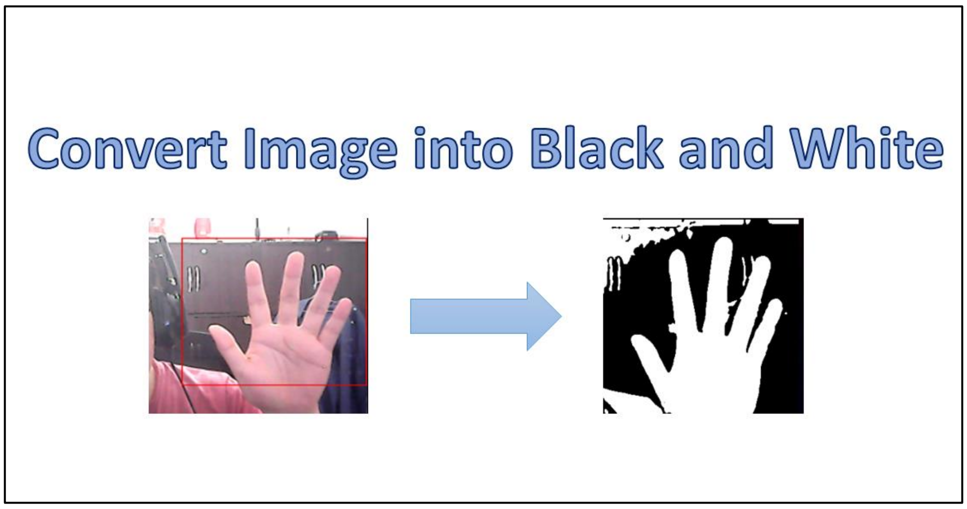


Figure 4.2: Convert Image into Black and White

After having the image with pure black and white the next step is to detect the convexity defects points from the hand shape. The first step for this phase is to determine the contours of the hand using OpenCV function “FindContour()” and use the contours value as the input in the “ConvexHull()” function to get the convex hull value. The “ConvexityDefects()” function will use both contours and convex hull values to find all the convexity defects points for the hand. Each defect point contains 3 values which are the near, far and start value. Some of the convexity defects point is not relevant for the finger’s numbers detection and will be eliminated by finding the degree of between these values. Only the convexity defects point that has degree greater than 100 will remain and be counted. When the number of convexity defects is 1 means there are 2 fingers, therefore this method can determine the numbers of fingers from 2 to 5. Figure 4.3 shows the meaning of contour, convex hull and convexity defect while Figure 4.4 shows the complete process for finding the required convexity defect points. The steps for the whole image detection process will be listed in Table 4-1.

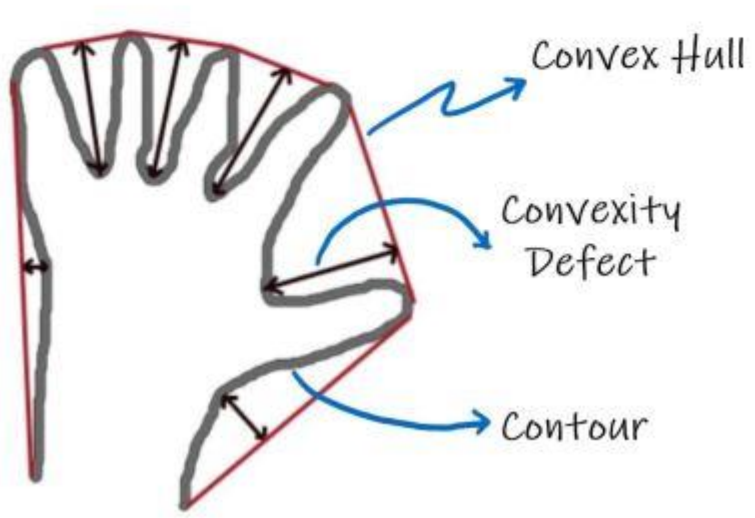


Figure 4.3: Meaning for Contour, Convex Hull and Convexity Defect

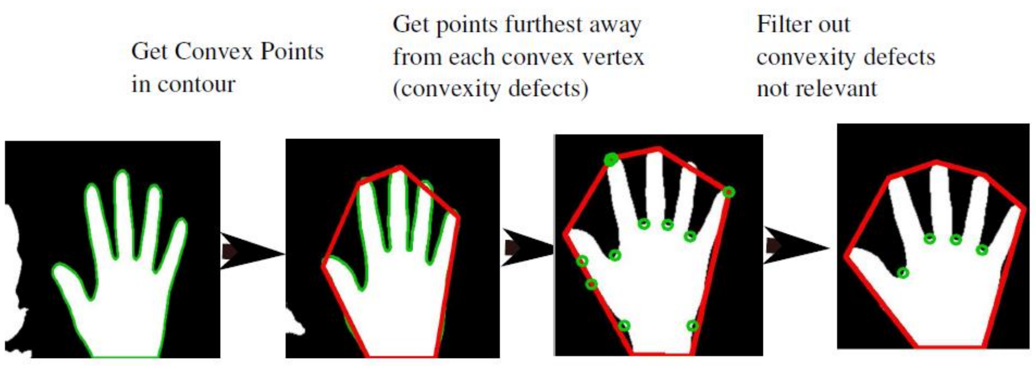

Get Convex Points in contour
Get points furthest away from each convex vertex (convexity defects)
Filter out convexity defects not relevant

Figure 4.4: Process for Finding Convexity Defects Points

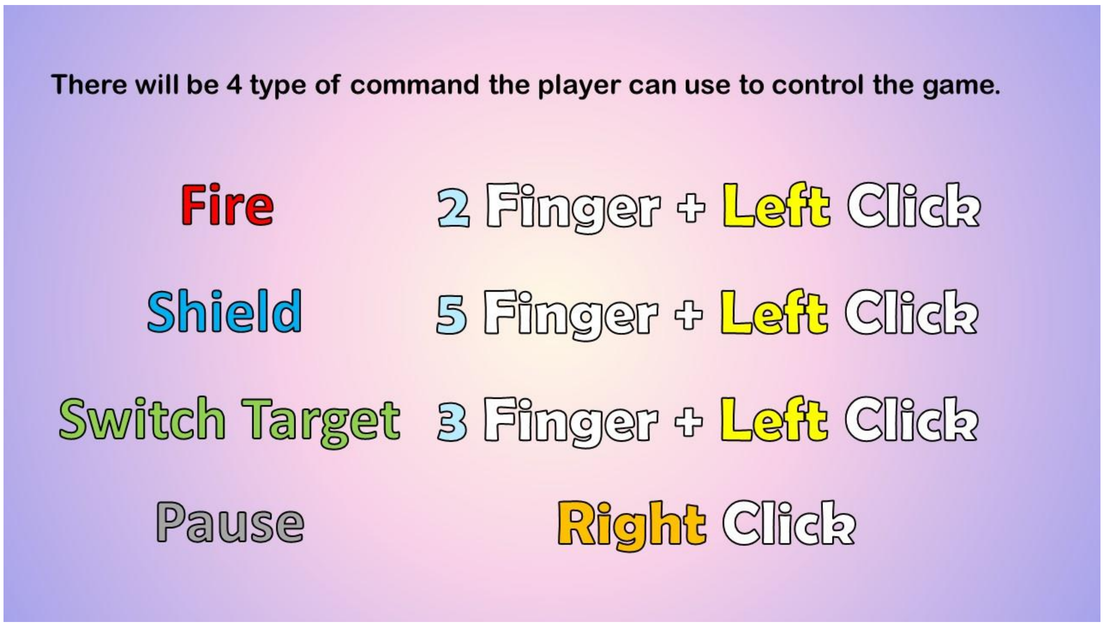
Table 4-1: Steps for Image Detection

1. Get ROI from Webcam. 1.1 Retrieve image data from webcam. 1.2 Retrieve ROI image value by extracting the data from specific location on the webcam image.

```
2. Convert the ROI image to black and white.
    2.1 Convert the image into grayscale image.
    2.2 Blur the image using Gaussian blur.
    2.3 Convert the image into black and white using threshold value.

3. Determine the numbers of convexity defects point.
    3.1 Find contours from black and white image.
    3.2 Find convex hull from contours.
    3.3 Find convexity defects from contours and convex hull.
    3.4 Filter convexity defects which is irrelevant.
    3.5 Count the numbers of convexity defects.
```

4.3.2 FPS Gaming Algorithm

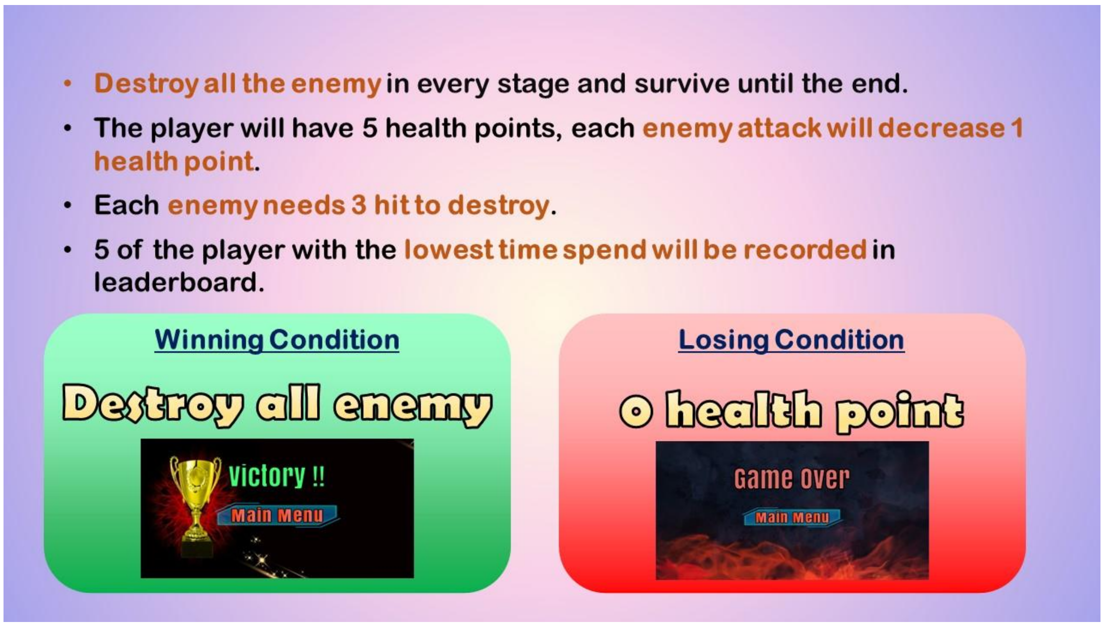
The FPS Game created in the completed system has a simple gaming process. Each stage will have three areas with 3 to 4 targets with the name “Drone” staying in the 3D space. These “Drone” will deal damage to the player after an interval of times. All the player needs to do is use the hand gesture and mouse click to deal damage to the “Drone” or block the incoming attack from the it. Figure 4.5 shows the input method for all 4 interactions in the FPS game. The player will be able to move to the next area after all enemy in the area has been destroyed. Figure 4.6 shows the winning and losing condition while Figure 4.7 shows flow of the game.



There will be 4 type of command the player can use to control the game.

Fire	2 Finger + Left Click
Shield	5 Finger + Left Click
Switch Target	3 Finger + Left Click
Pause	Right Click

Figure 4.5: Input Method for the 4 Interaction in the FPS Game



- Destroy all the enemy in every stage and survive until the end.
- The player will have 5 health points, each enemy attack will decrease 1 health point.
- Each enemy needs 3 hit to destroy.
- 5 of the player with the lowest time spend will be recorded in leaderboard.

Winning Condition

Destroy all enemy

Losing Condition

0 health point

Victory !!
Main Menu
Game Over
Main Menu

Figure 4.6: Winning and Losing Condition

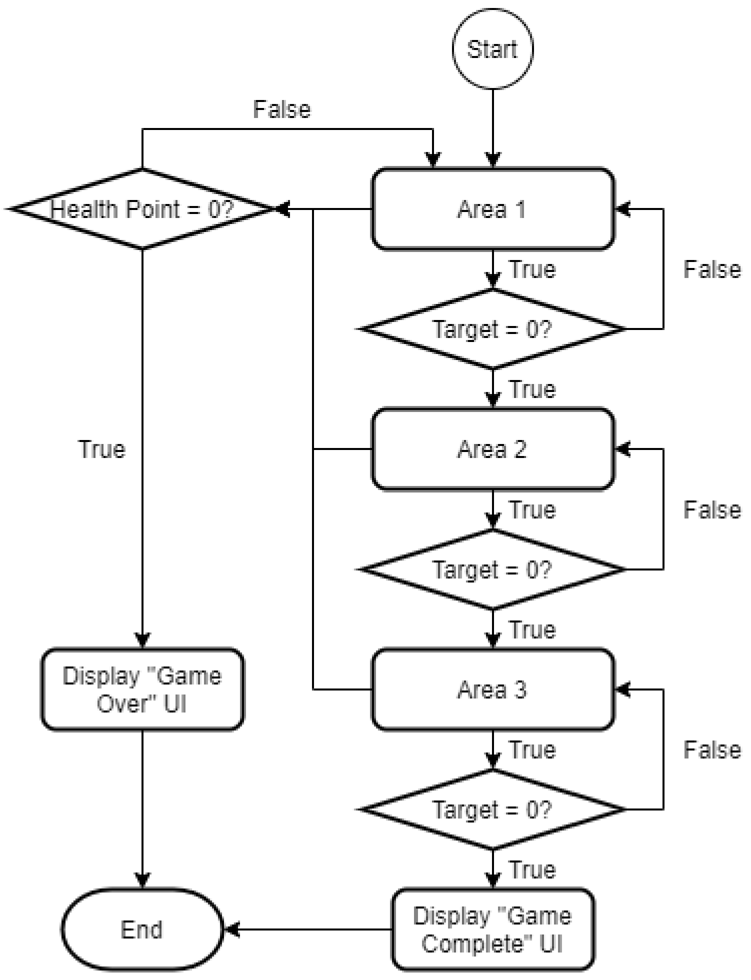


Figure 4.7: Flow of the FPS Game

4.4 Scene Design in Unity

There will be 6 scenes inside Unity3D with the name, “Information”, “Login”, “Main Menu”, “Easy Stage”, “Normal Stage” and “Hard Stage”. These 6 scenes can be categorized into 2 main parts, the “User Interface” and “FPS Game” as shown in Figure 4.8.

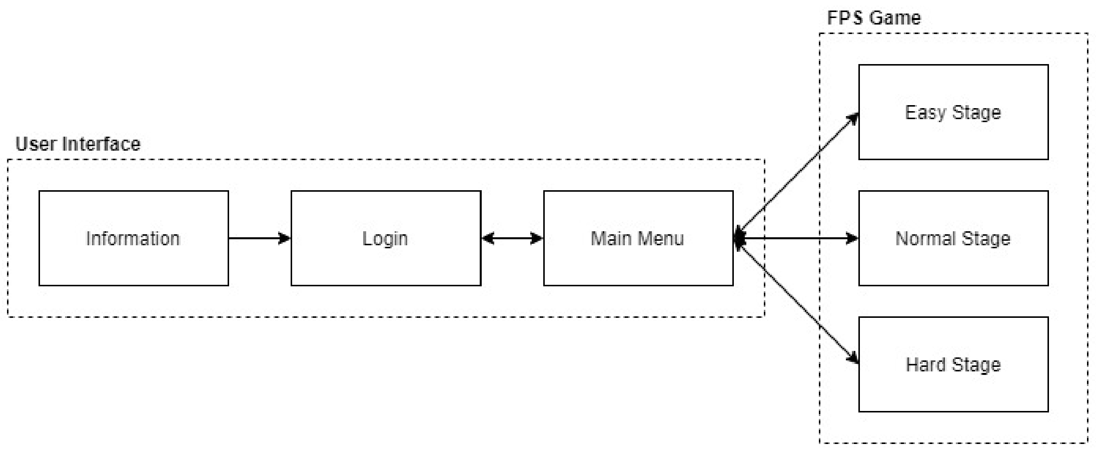


Figure 4.8: Scene in Unity

4.4.1 “Information” Scene

The “Information” scene is a one-page user interface which provides basic information about the game. The user only needs to click on the “Confirm” button if they wish to continue or “Quit” button to quit the game. If the user has click on the “Confirm” button, the scene will be redirected to the “Login” scene.

Figure 4.9 shows the C# script contains in “Information” scene. This scene only needs one C# script with name “Information Manager” that use to manage the scene change when the user clicks on the button.

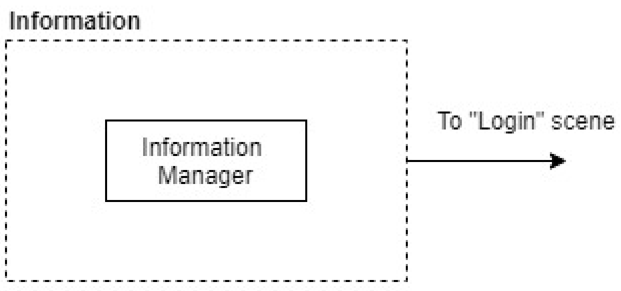


Figure 4.9: "Information" Scene Design

4.4.2 “Login” Scene

The register and login function will be provided in this scene. New user can create a new account by input the username and password in the register section. They will be able to redirect to the “Main Menu” scene if their username does not exist in the Playfab database. Those users who did register before their account can insert their username and password in the login section and proceed to the “Main Menu” scene.

Figure 4.10 shows the C# scripts inside the “Login Scene”. “Game Default Setup Manager” and “Playfab Manager” is the scripts which will existed throughout the game without getting destroy while the scene is changing. “Game Default Setup Manager” stores the value that has been setup by the user in the setting section while “Playfab Manager” is the scripts use to connect, send, and retrieve user information from the Playfab database. “Playfab Manager” will store the username after the login has been completed. “Login Manager” only contains the function use to close the application when the user has click on the “Quit” button.

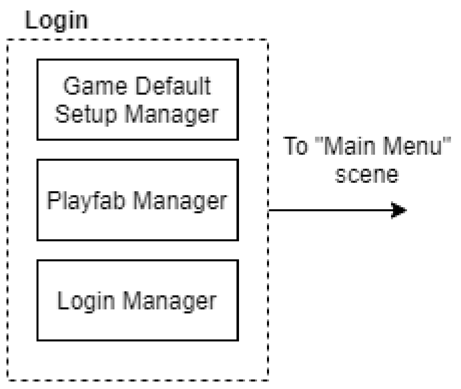


Figure 4.10: "Login" Scene Design

4.4.3 “Main Menu” Scene

“Main Menu” scene provides variety of selection for the user, the user will be able start the game by choosing the difficulty, setting the image detection preference, view the leaderboard, study the tutorial or logout their account at this scene. Each kind of function will have its own sub section in this scene with different interface design.

Figure 4.11 shows the design for “Main Menu” scene, the user will be able to swap between each user interface by clicking the specific button in the scene. The “Main Menu Manager” is the C# script that contains all the function for the buttons in this scene and manage the change of user interface. The setting user interface and tutorial user interface has its own manager because these two interfaces contain more interaction and having its own script will decrease the complexity of the function in “Main Menu Manager”.

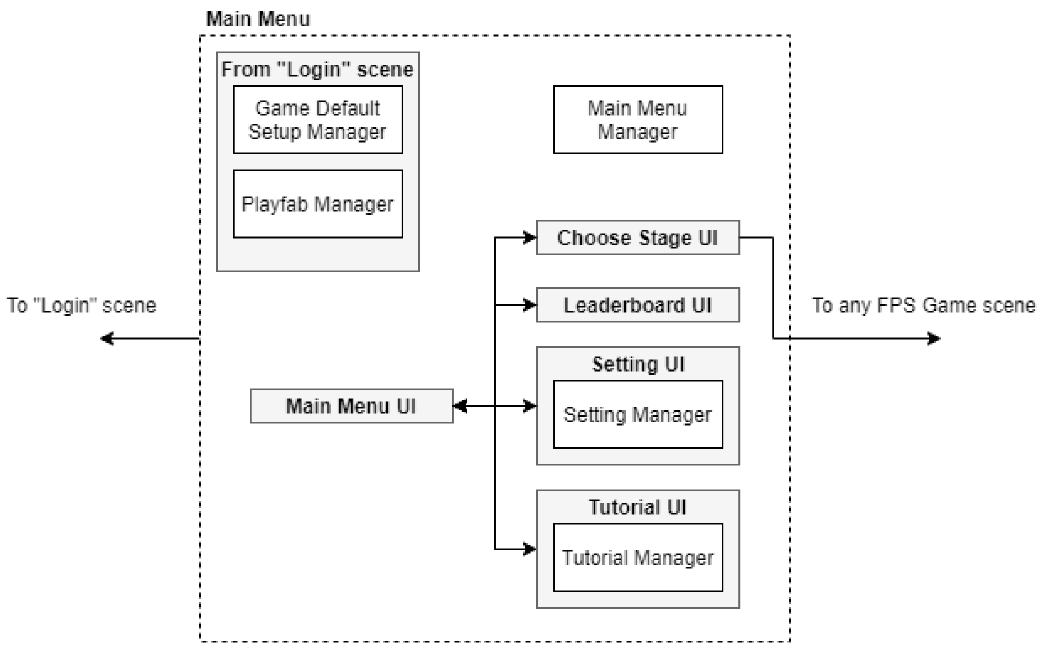


Figure 4.11: "Main Menu" Scene Design

4.4.4 FPS Game Scene

The "Easy Stage", "Normal Stage" and "Hard Stage" scenes are identical with only the change in the fire interval of the enemy, the higher the difficulty, the shorter the fire interval. Therefore, all 3 scenes can be categorized under the FPS Game Scene since the C# script and game object in the scene are almost the same.

The design of the FPS Game scene has been shown in Figure 4.12 with the "FPS Game Manager" acting as the primary script that determines the condition of the game to change the user interface depending on the situation. The "Image Detection" script contains all the functions related to the image detection process that are used to determine the hand gesture received from the webcam and sent to another manager which needs it to work on the output in the FPS game.

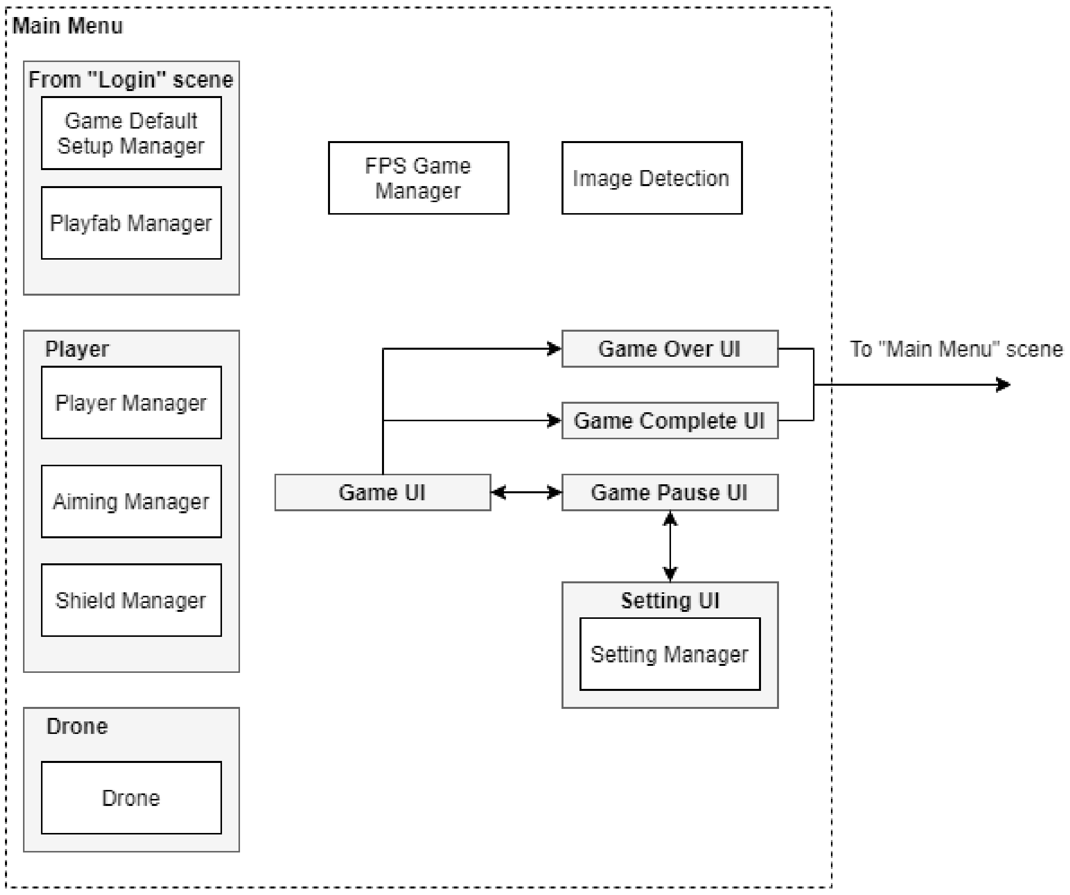


Figure 4.12: FPS Game Scene Design

The “Player” game object has 3 C# scripts embedded on it with names “Player Manager”, “Shield Manager” and “Aiming Manager”. “Player Manager” stores the value of the player health point while the “Shield Manager” determines the condition of the shield which will be generated by the player using a hand gesture. “Aiming Manager” is needed to auto-trace the position of the enemy which is the “Drone” in the scene and will automatically move the scope on it when the previous enemy is destroyed, or the player has used the hand gesture to switch targets.

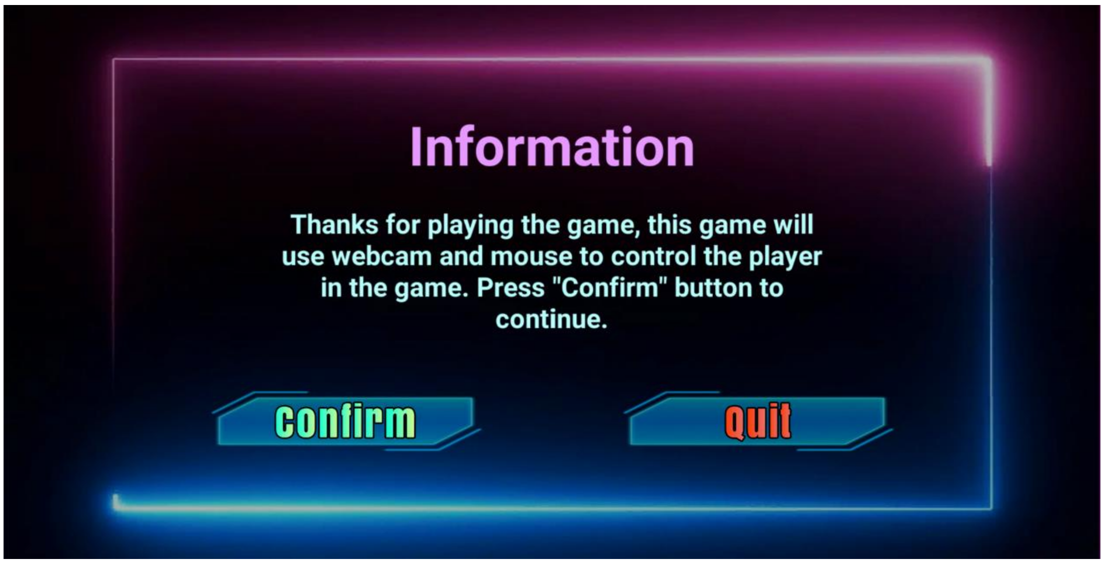
The “Game UI” is the user interface for the gaming process and can be switched to “Game Pause UI” when the player decides to pause the game. Then they will be able to go to the “Setting UI” which has the same interface as in the “Main Menu” scene where they can change the threshold value for the image detection. The “Game Over UI” and “Game Complete UI” will be automatically redirected from “Game UI” depending on the condition of the game. When the player health point reaches 0, then “Game Over UI” will be activated while all the enemy has been destroyed will generate the “Game Complete UI”.

4.5 Interface Design

The explanation of the interface design will be separated into 4 sub sections depending on the scene with the name of the scene. There will be further explanation about each element of the object in the interface for the FPS Game.

4.5.1 “Information” Interface

This interface will inform the player that the application will need them to have a webcam and mouse as the input of the game as shown in Figure 4.13. The player can either click on the “Confirm” button to continue or “Quit” button to close the application.


Information

Thanks for playing the game, this game will use webcam and mouse to control the player in the game. Press "Confirm" button to continue.

Figure 4.13: "Information" Interface

4.5.2 "Login" Interface

In the "Login" Interface, the player is able to login their account by inputting the username and password. New players are able to register their account by filling in their username and password in the register field. The user will be able to instantly login and go to the Main Menu Interface if they register successfully. The username is the primary key for each user, therefore any username input that has existed in the Playfab player database will be rejected during the registration. Figure 4.14 shows the "Login" Interface.

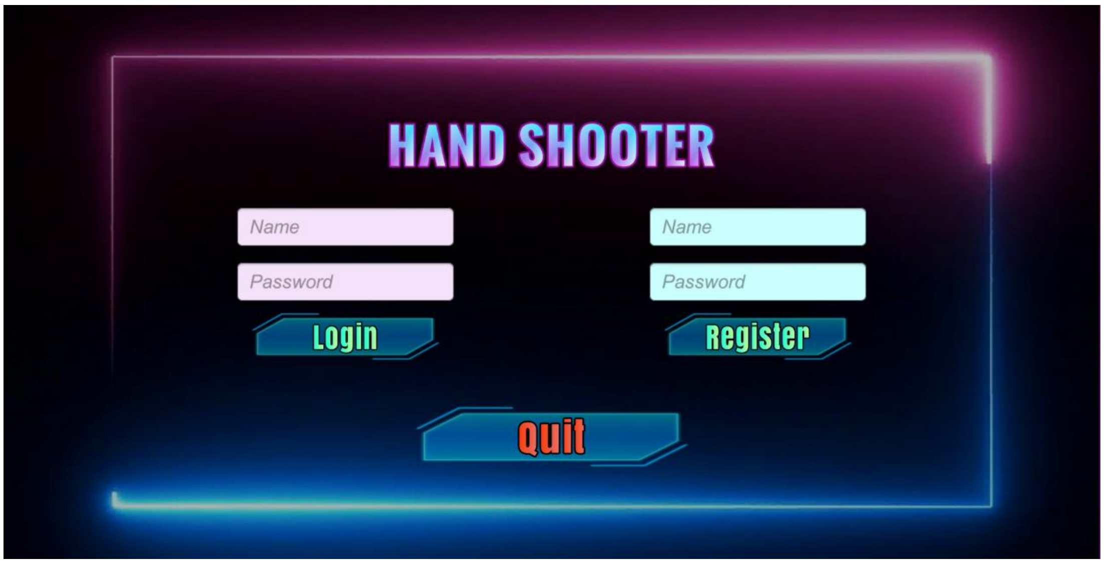


Figure 4.14: Login Interface

4.5.3 “Main Menu” Interface

There will be 6 selection buttons for the “Main Menu” interface, which is “Start Game”, “Choose Stage”, “Tutorial”, “Leaderboard”, “Logout” and “Tutorial” as shown in Figure 4.15. Each button will change the user interface displayed to the player.

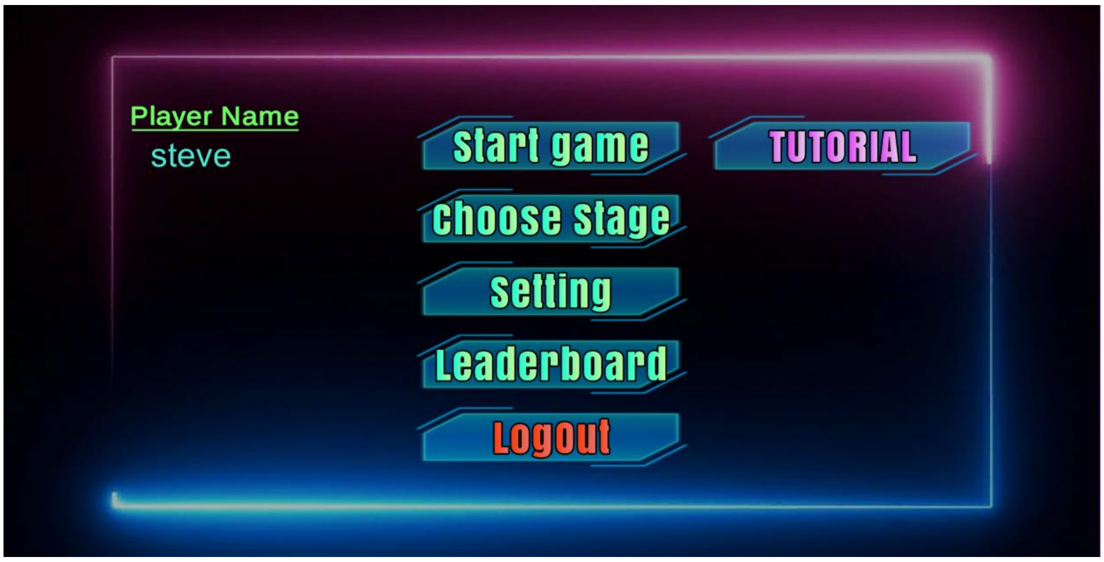


Figure 4.15: “Main Menu” Interface

4.5.3.1 “Choose Stage” Interface

The player will be able to start the FPS game in Easy stage when pressing the “Start Game” button or they can choose the difficulty of the stage by pressing the “Choose Stage” button. Figure 4.16 is the user interface when the player clicks on the “Choose Stage” button.

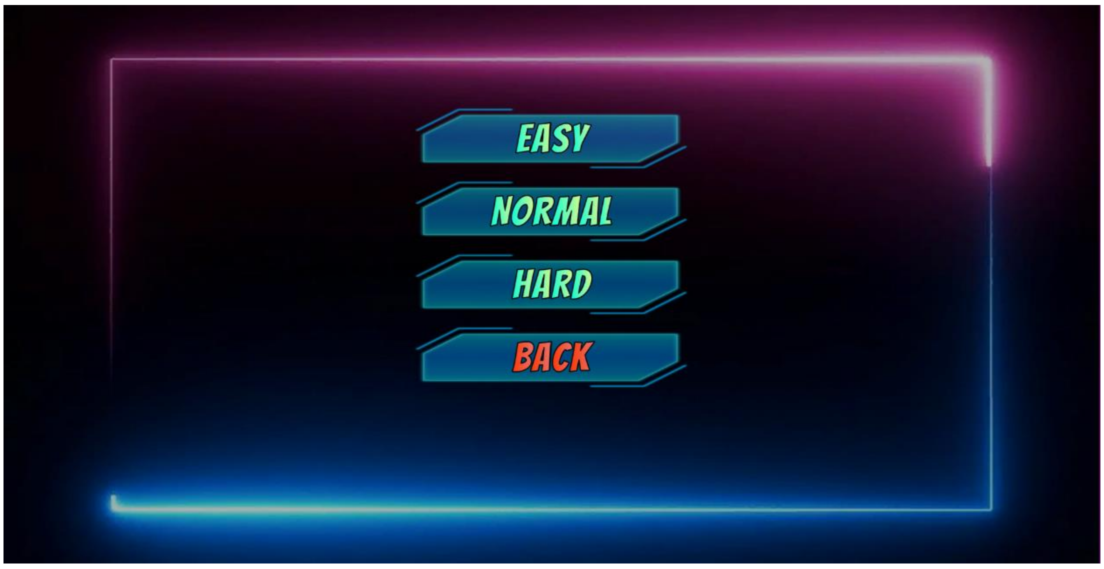


Figure 4.16: "Choose Stage" Interface

4.5.3.2 “Setting” Interface

The “Setting” button will change the user interface displayed from “Login” interface to “Setting” interface. The player will be able to change the threshold value for the image detection script to the value that suits their webcam resolution to ensure the image detection works fine depending on their environment. Figure 4.17 and 4.18 shows the sample user interface before and after changing the threshold value in the “Setting” interface.

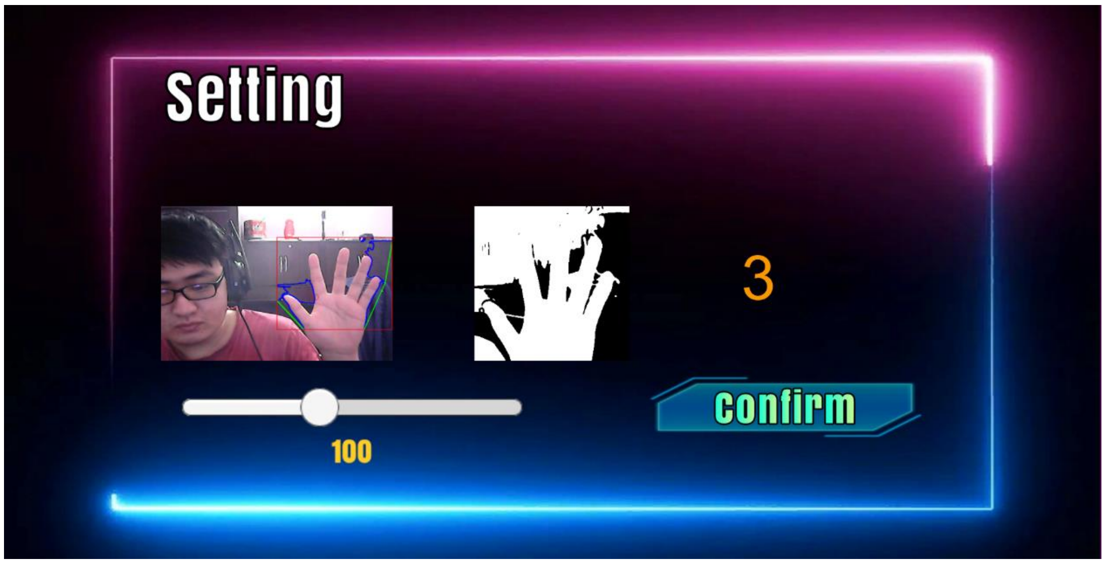


Figure 4.17: “Setting” Interface (Before)

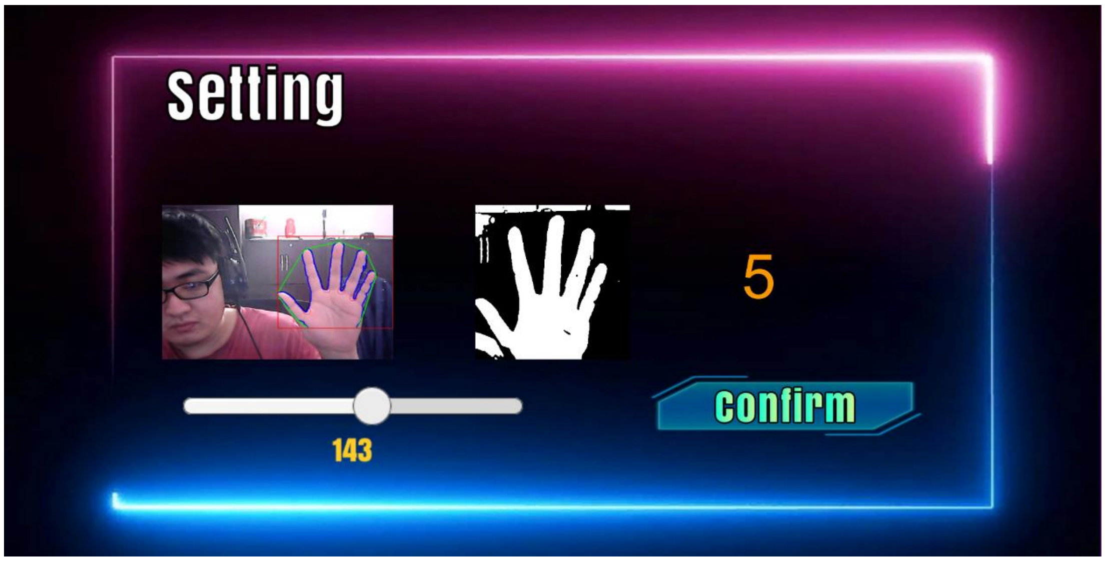
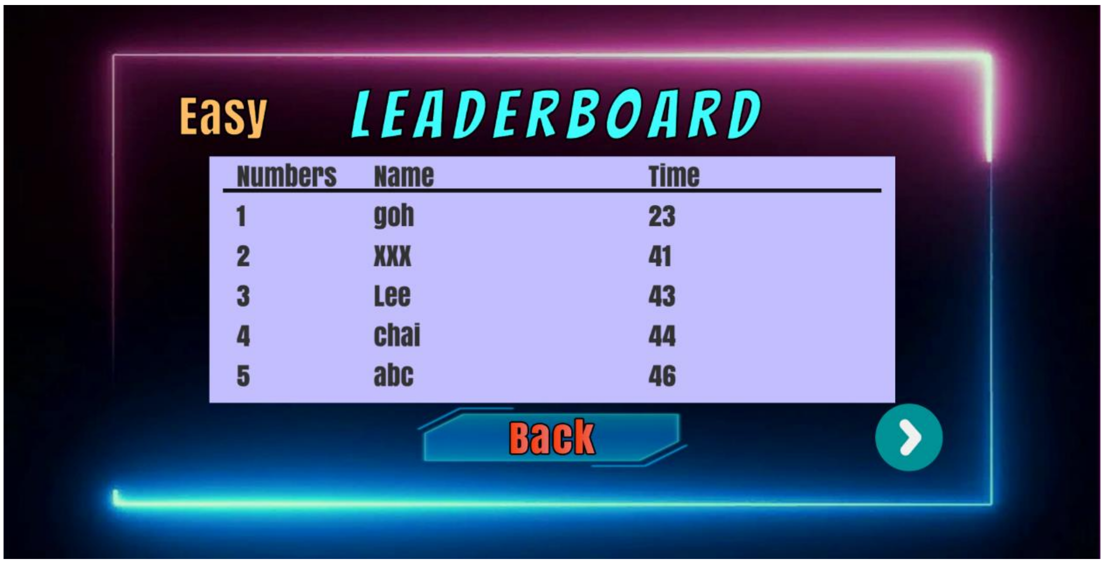


Figure 4.18: “Setting” Interface (after)

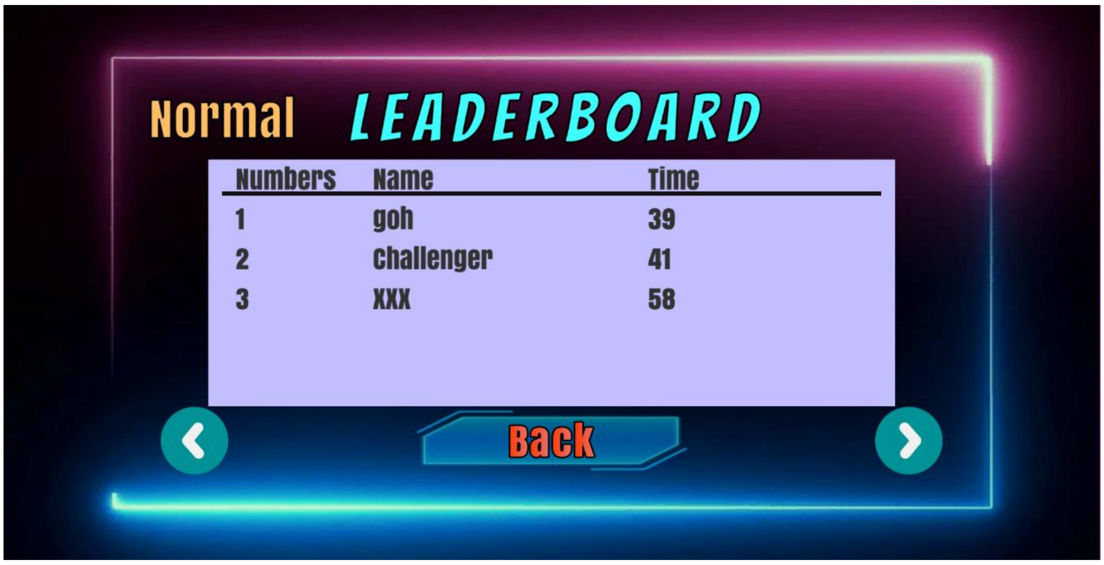
4.5.3.3 “Leaderboard” Interface

The “Leaderboard” interface contains 3 changeable pages for all 3 difficulties. The player can click on the button with arrow icon to switch between the leaderboard pages or click on the “Back” button to return to “Main Menu” interface. Figures 4.19, 4.20 and 4.21 show all 3 interfaces for the leaderboard.



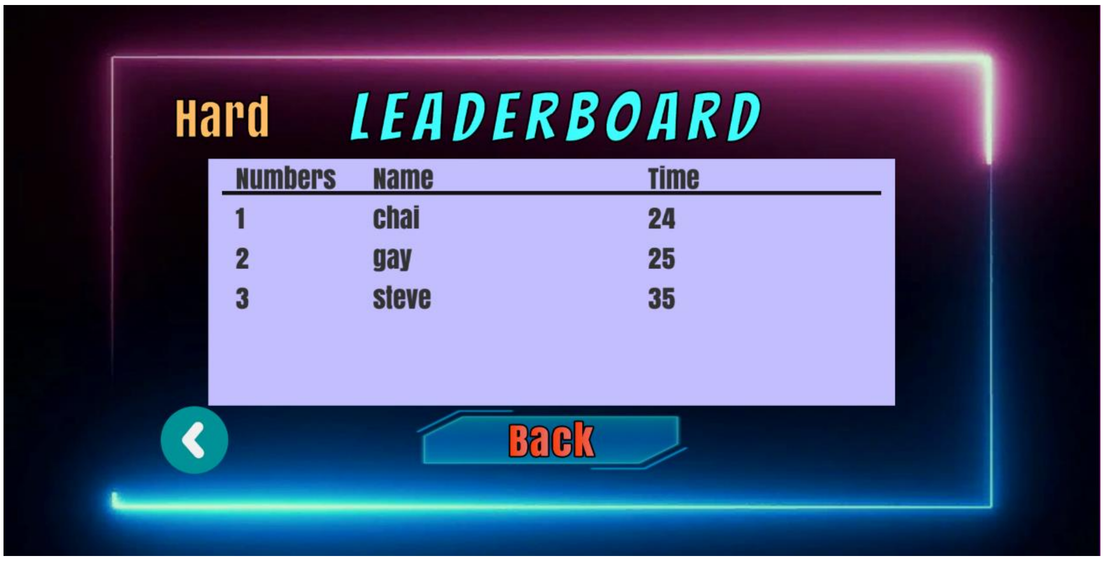
Numbers	Name	Time
1	goh	23
2	XXX	41
3	Lee	43
4	chai	44
5	abc	46

Figure 4.19: “Easy Leaderboard” Interface



Numbers	Name	Time
1	goh	39
2	Challenger	41
3	XXX	58

Figure 4.20: "Normal Leaderboard" Interface

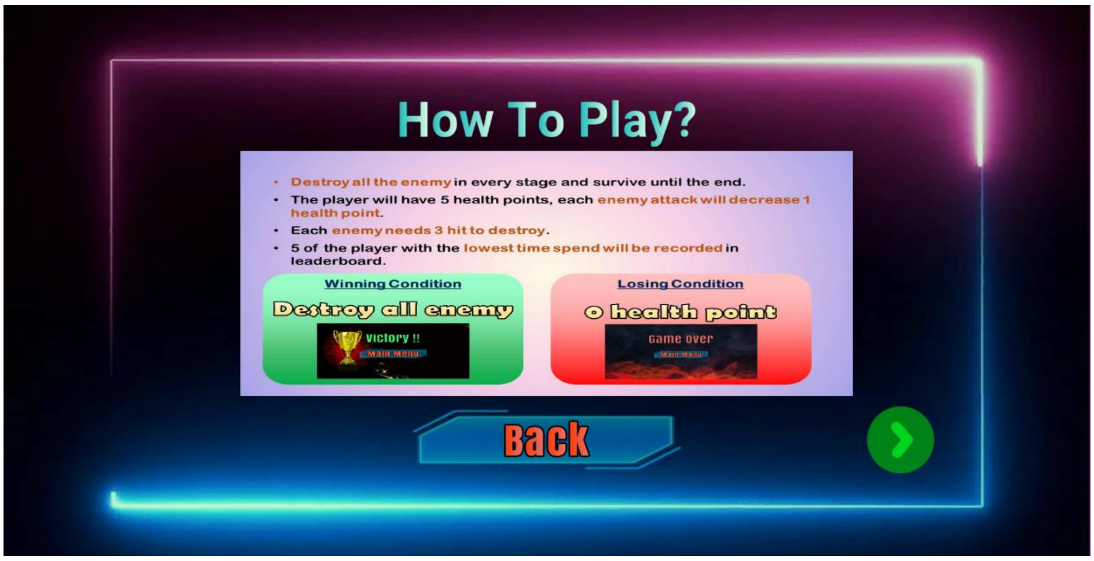


Numbers	Name	Time
1	chai	24
2	gay	25
3	steve	35

Figure 4.21: "Hard Leaderboard" Interface

4.5.3.4 “Tutorial” Interface

The “Tutorial” Interface contains all the information that the player needs to understand about the gameplay of the FPS game. There will be 5 pages provided in this interface as shown in Figure 4.22, 4.23, 4.24, 4.25 and 4.26. Similar to the "Leaderboard" interface, the page can be changed using the button with an arrow icon.



How To Play?

- Destroy all the enemy in every stage and survive until the end.
- The player will have 5 health points, each enemy attack will decrease 1 health point.
- Each enemy needs 3 hit to destroy.
- 5 of the player with the lowest time spend will be recorded in leaderboard.

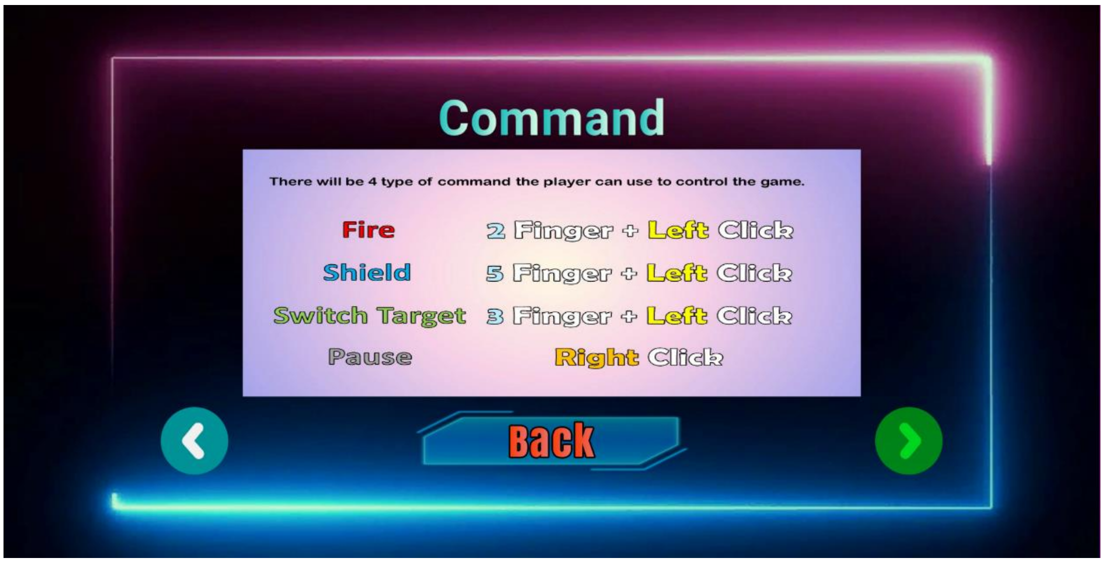
Winning Condition

Destroy all enemy

Losing Condition

0 health point

Figure 4.22: "Tutorial" Interface 1



Command

There will be 4 type of command the player can use to control the game.

Fire	2 Finger + Left Click
Shield	5 Finger + Left Click
Switch Target	3 Finger + Left Click
Pause	Right Click

Figure 4.23: "Tutorial" Interface 2

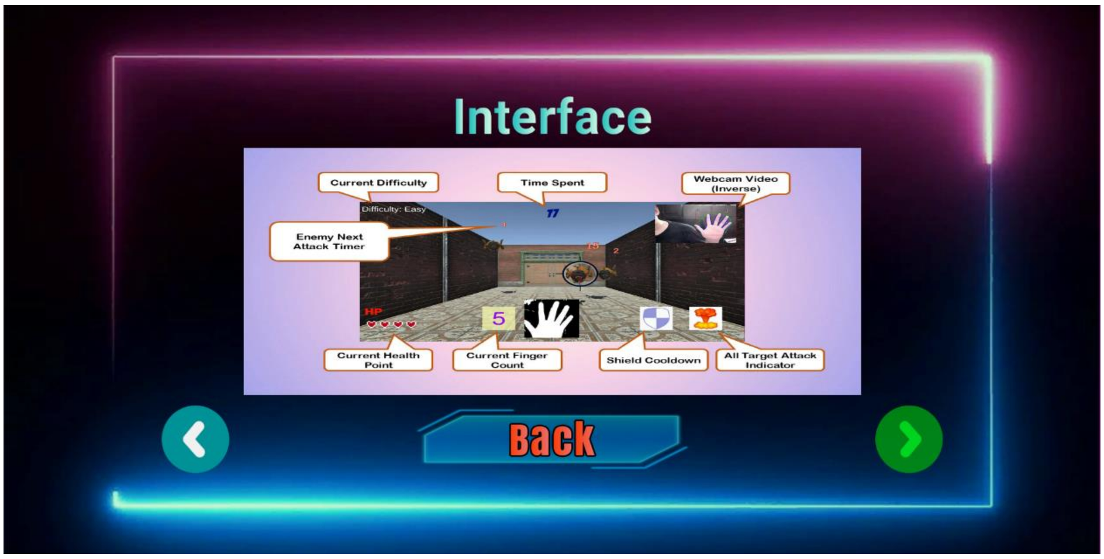
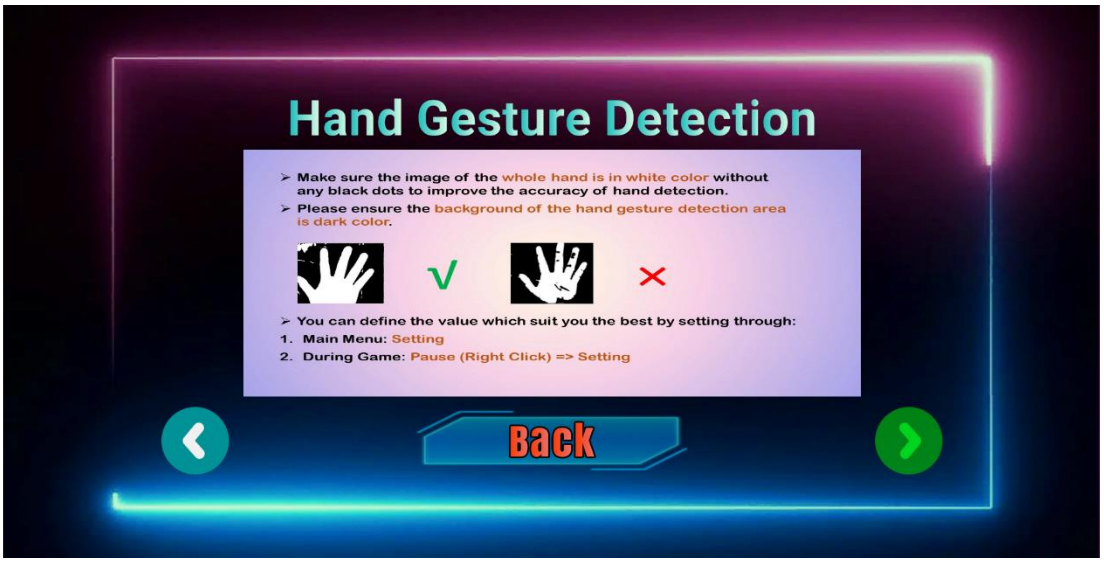


Figure 4.24: "Tutorial" Interface 3



Hand Gesture Detection

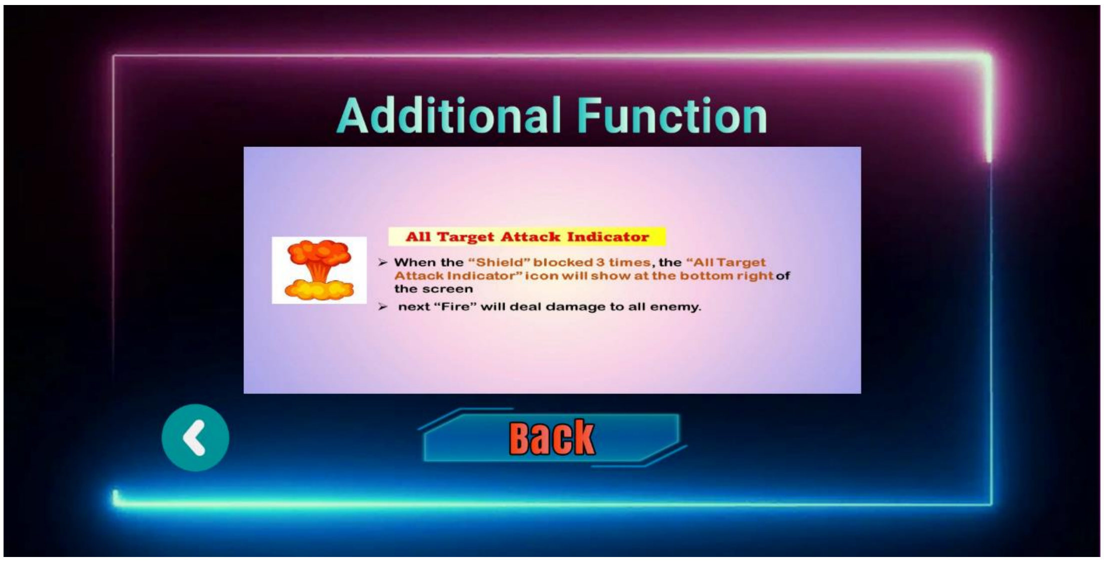
- Make sure the image of the whole hand is in white color without any black dots to improve the accuracy of hand detection.
- Please ensure the background of the hand gesture detection area is dark color.

√ ×

- You can define the value which suit you the best by setting through:

1. Main Menu: Setting
2. During Game: Pause (Right Click) => Setting

Figure 4.25: "Tutorial" Interface 4



Additional Function

All Target Attack Indicator

- When the “Shield” blocked 3 times, the “All Target Attack Indicator” icon will show at the bottom right of the screen
- next “Fire” will deal damage to all enemy.

Figure 4.26: "Tutorial" Interface 5

4.5.4 FPS Game Interface

Figure 4.27 shows the FPS Game Interface with labels. Label 1 is the text element which informs the player about the difficulty of the current stage. Label 2 is the text element which shows the total time the FPS game has gone through. The shorter the time to complete the FPS game, the higher the ranking the player will be on the leaderboard. Label 3 is the real-time video captured from the player webcam so that the player can know where their hands is and move their hand to the detection area.

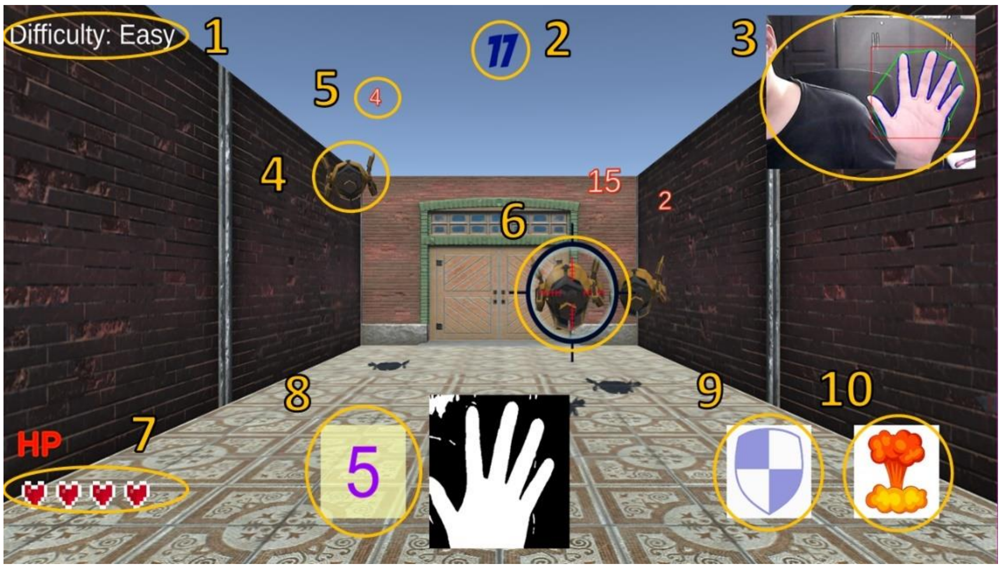


Figure 4.27: FPS Game Interface

Label 4 is an object which represents the enemy. The player needs to take down all of the enemies in the scene to proceed to the next area. Label 5 is the countdown value of the next attack timing of the enemy. Label 6 is the scope that is used to inform the player which is the enemy they are currently targeting.

Label 7 is the remaining health points of the player. The health points start with 5 and when the health points go to 0, the FPS game will be over. Label 8 is the number of fingers detected from the detection area. This field will only display five types of values which are 2, 3, 4, 5 for the finger counts and X for no detection.

Label 9 will show the availability of the "Shield" command in the game. It will count down from 5 to 0 when the player uses the shield, and the shield will be temporarily unavailable until the countdown ends. Label 10 is the "All Target Attack Indicator" icon. When the player is able to block 3 attacks from the enemy, this icon will show up and the next player attack will deal to all enemies.

There will be 3 types of interfaces that the FPS Game will be provided depending on the situation. Figure 4.28 is the "Game Pause" interface, the player will see this interface when they have right clicked using the mouse to temporarily stop the gaming process and they can go to the “Setting” interface using this interface. The player will see the “Game Over” interface as shown in Figure 4.29 if their health points go to 0 or the “Game Complete” interface as shown in Figure 4.30 if they have successfully destroyed all the enemies in the stage.

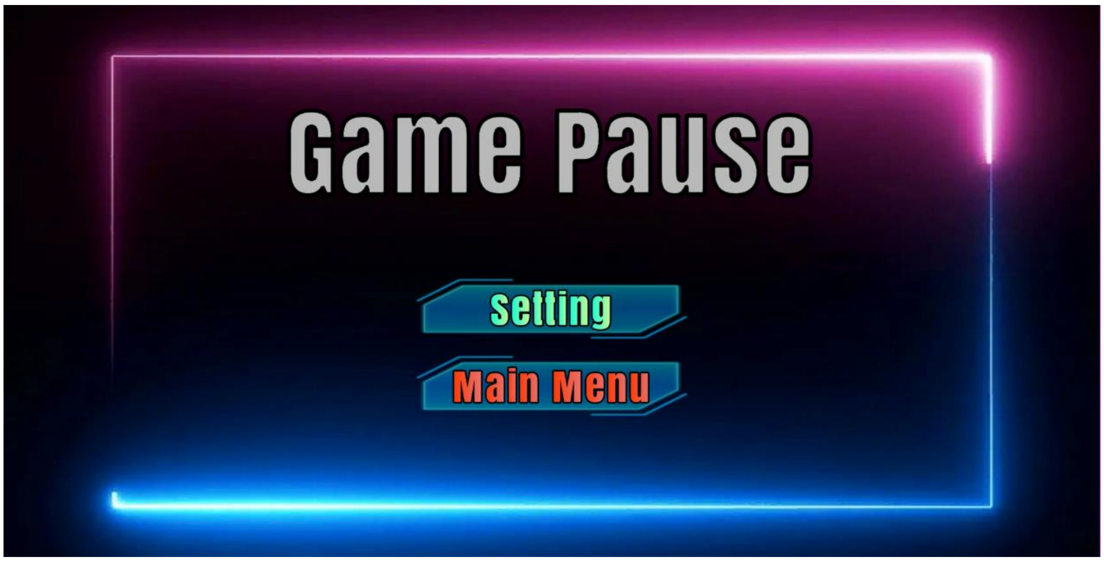


Figure 4.28: "Game Pause" Interface

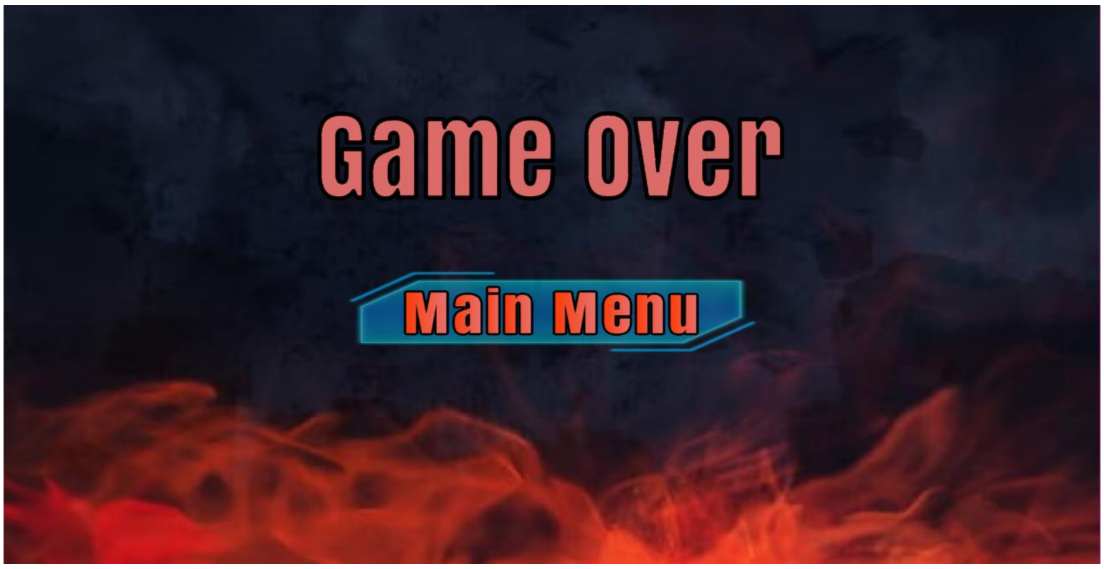


Figure 4.29: "Game Over " Interface

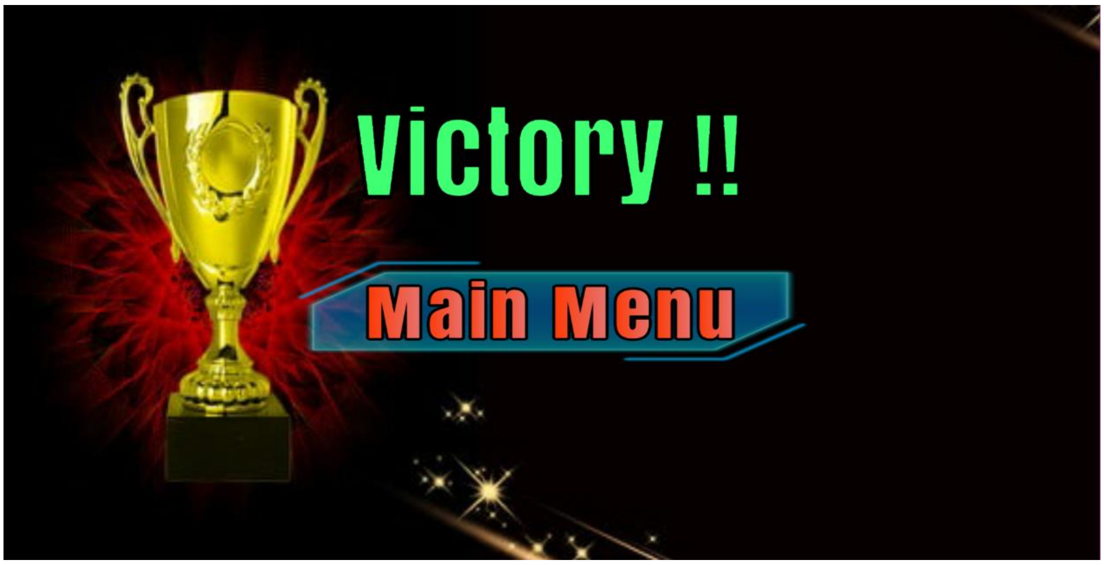


Figure 4.30: "Game Complete" Interface

4.6 Chapter Summary

Subchapter 4.2 listed down those functional requirements and non-functional requirements that have been fulfilled by the completed system.

Subchapter 4.3 describes the components that were used to construct the completed system. These components are Unity3D, OpenCV+ Unity and Microsoft Azure Playfab. The FPS gaming algorithm and image detection algorithm will also be included in this subchapter.

The design for the scene in Unity will be explained in subchapter 4.4. There will be 6 scenes for the system with the names “Information”, “Login”, “Main Menu”, “Easy Stage”, “Normal Stage” and “Hard Stage”. The last 3 scenes will be described together in the FPS Game Scene section since all 3 scenes have a similar environment with minor modification.

Subchapter 4.5 shows and explains about the interface design of the completed system following the numbers of scenes in Unity.

CHAPTER 5

IMPLEMENTATION AND TESTING

5.1 Introduction

This chapter will discuss about the coding of the system and the results get from the program. The testing result will also be described in this chapter.

5.2 Coding of system's main function

The coding that uses in the whole system is C# programming language. Unity provides the feature that the C# script can be embedded to certain GameObject in the scene in order to define the behavior of the system at certain scene. The 4 primary C# script setups in this project is "Main Menu Manager", "PlayFab Manager", "Game Default Setup Manager" and "FPS Game Manager". Figure 5.1 shows the main C# script for the whole system.

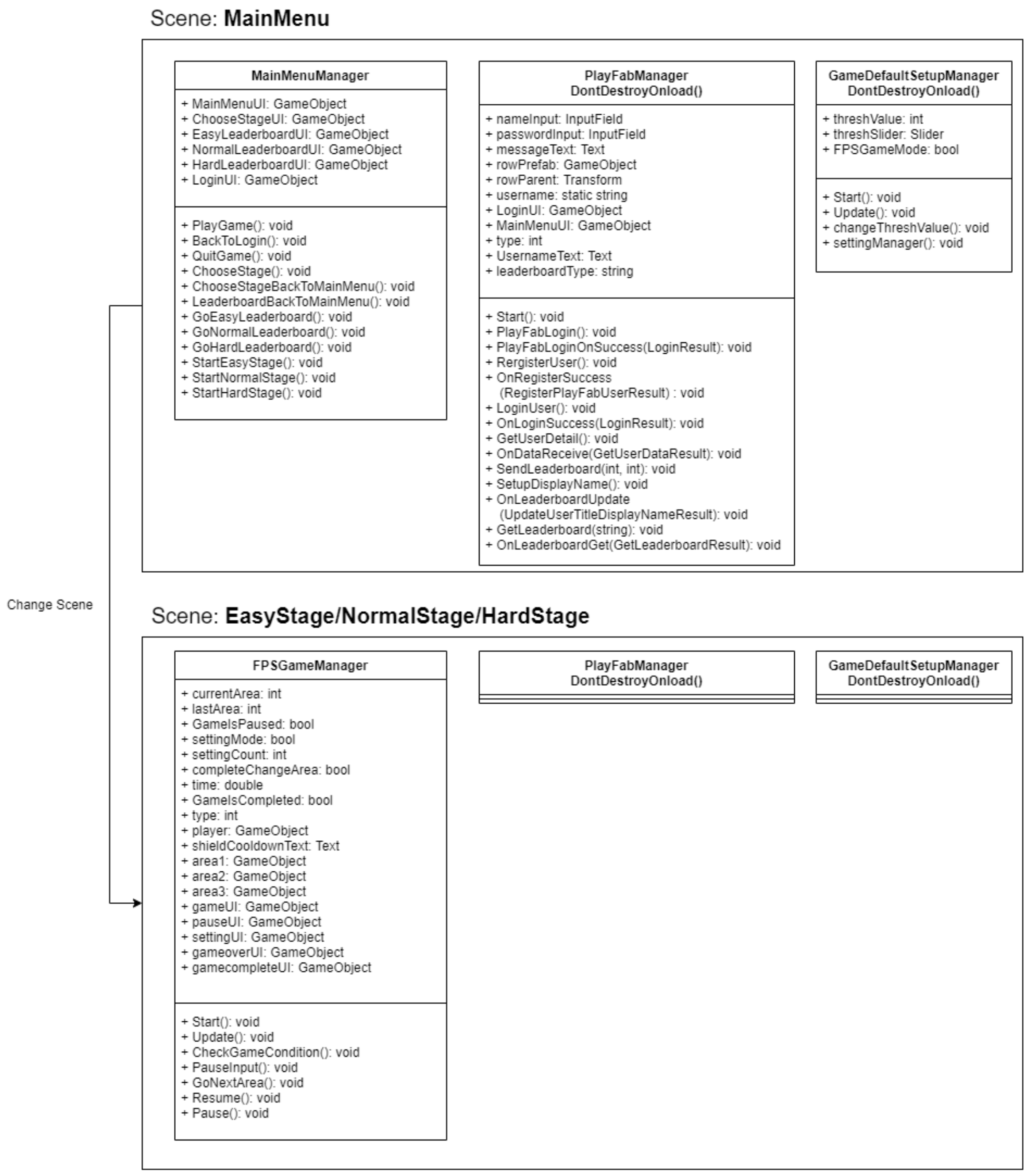
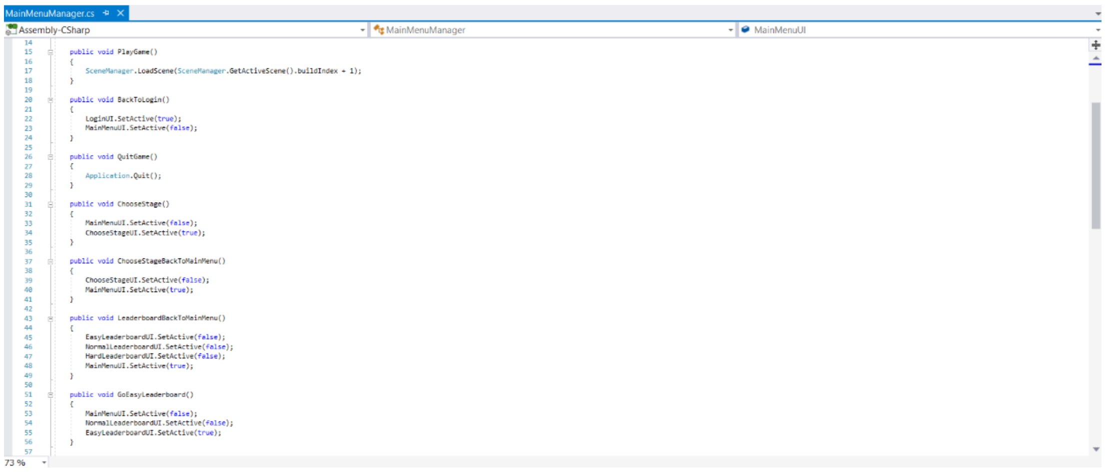

Scene: MainMenu
MainMenuManager
+ MainMenuUI: GameObject
+ ChooseStageUI: GameObject
+ EasyLeaderboardUI: GameObject
+ NormalLeaderboardUI: GameObject
+ HardLeaderboardUI: GameObject
+ LoginUI: GameObject
+ PlayGame(): void
+ BackToLogin(): void
+ QuitGame(): void
+ ChooseStage(): void
+ ChooseStageBackToMainMenu(): void
+ LeaderboardBackToMainMenu(): void
+ GoEasyLeaderboard(): void
+ GoNormalLeaderboard(): void
+ GoHardLeaderboard(): void
+ StartEasyStage(): void
+ StartNormalStage(): void
+ StartHardStage(): void
PlayFabManager DontDestroyOnload()
+ nameInput: InputField
+ passwordInput: InputField
+ messageText: Text
+ rowPrefab: GameObject
+ rowParent: Transform
+ username: static string
+ LoginUI: GameObject
+ MainMenuUI: GameObject
+ type: int
+ UsernameText: Text
+ leaderboardType: string
+ Start(): void
+ PlayFabLogin(): void
+ PlayFabLoginOnSuccess(LoginResult): void
+ RergisterUser(): void
+ OnRegisterSuccess (RegisterPlayFabUserResult) : void
+ LoginUser(): void
+ OnLoginSuccess(LoginResult): void
+ GetUserDetail(): void
+ OnDataReceive(GetUserDataResult): void
+ SendLeaderboard(int, int): void
+ SetupDisplayName(): void
+ OnLeaderboardUpdate (UpdateUserTitleDisplayNameResult): void
+ GetLeaderboard(string): void
+ OnLeaderboardGet(GetLeaderboardResult): void
GameDefaultSetupManager DontDestroyOnload()
+ threshValue: int
+ threshSlider: Slider
+ FPSGameMode: bool
+ Start(): void
+ Update(): void
+ changeThreshValue(): void
+ settingManager(): void
Change Scene
Scene: EasyStage/NormalStage/HardStage
FPSGameManager
+ currentArea: int
+ lastArea: int
+ GameIsPaused: bool
+ settingMode: bool
+ settingCount: int
+ completeChangeArea: bool
+ time: double
+ GameIsCompleted: bool
+ type: int
+ player: GameObject
+ shieldCooldownText: Text
+ area1: GameObject
+ area2: GameObject
+ area3: GameObject
+ gameUI: GameObject
+ pauseUI: GameObject
+ settingUI: GameObject
+ gameoverUI: GameObject
+ gamecompleteUI: GameObject
+ Start(): void
+ Update(): void
+ CheckGameCondition(): void
+ PauseInput(): void
+ GoNextArea(): void
+ Resume(): void
+ Pause(): void
PlayFabManager DontDestroyOnload()
GameDefaultSetupManager DontDestroyOnload()

Figure 5.1: Main C# Script int the System

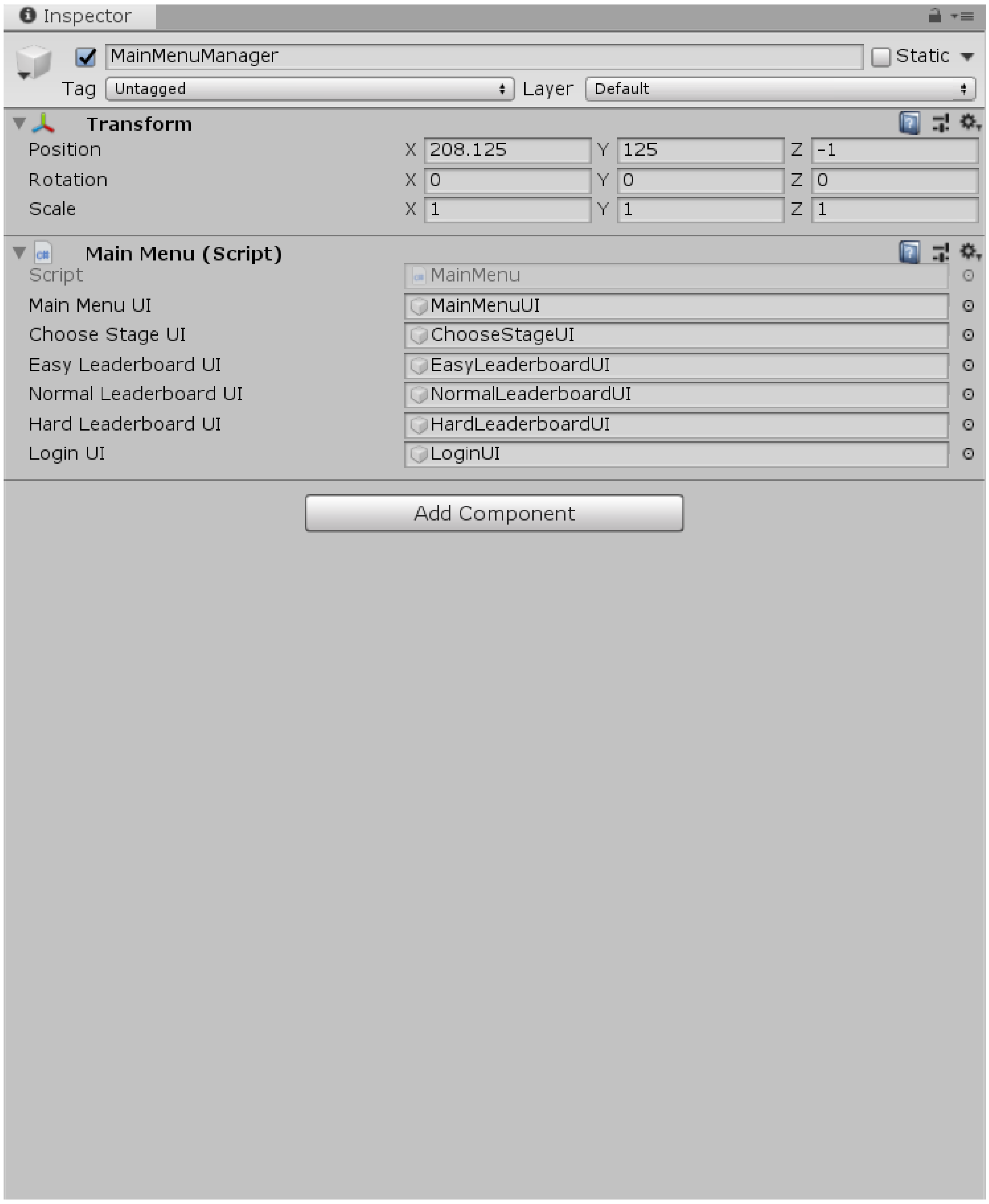
5.2.1 Main Menu Manager

This C# scripts determine the user interface that will be display according to the user selection. The first interface the user will be go through is “Login User Interface” which required the user to either login using existed username and password or register a new account for the system. “Main Menu Manager” contains all the function which will be used to switch between the user interface depending on the button that has been pressed by the user. The specific function in “Main Menu Manager” will be called by the OnClick() function provided in each button. Figure 5.2 shows the coding of “Main Menu Manager” while Figure 5.3 is the public variable which will be display by the inspector panel in Unity.



```

    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }

    public void BackToLogin()
    {
        LoginUI.SetActive(true);
        MainMenuUI.SetActive(false);
    }

    public void QuitGame()
    {
        Application.Quit();
    }

    public void ChooseStage()
    {
        MainMenuUI.SetActive(false);
        ChooseStageUI.SetActive(true);
    }

    public void ChooseStageBackToMainMenu()
    {
        ChooseStageUI.SetActive(false);
        MainMenuUI.SetActive(true);
    }

    public void LeaderboardBackToMainMenu()
    {
        EasyLeaderboardUI.SetActive(false);
        NormalLeaderboardUI.SetActive(false);
        HardLeaderboardUI.SetActive(false);
        MainMenuUI.SetActive(true);
    }

    public void GoEasyLeaderboard()
    {
        MainMenuUI.SetActive(false);
        NormalLeaderboardUI.SetActive(false);
        EasyLeaderboardUI.SetActive(true);
    }

```

Figure 5.2: Coding of "Main Menu Manager" script


Inspector

MainMenuManager — Static

Tag: Untagged — Layer: Default

Transform

Position	X	208.125	Y	125	Z	-1
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

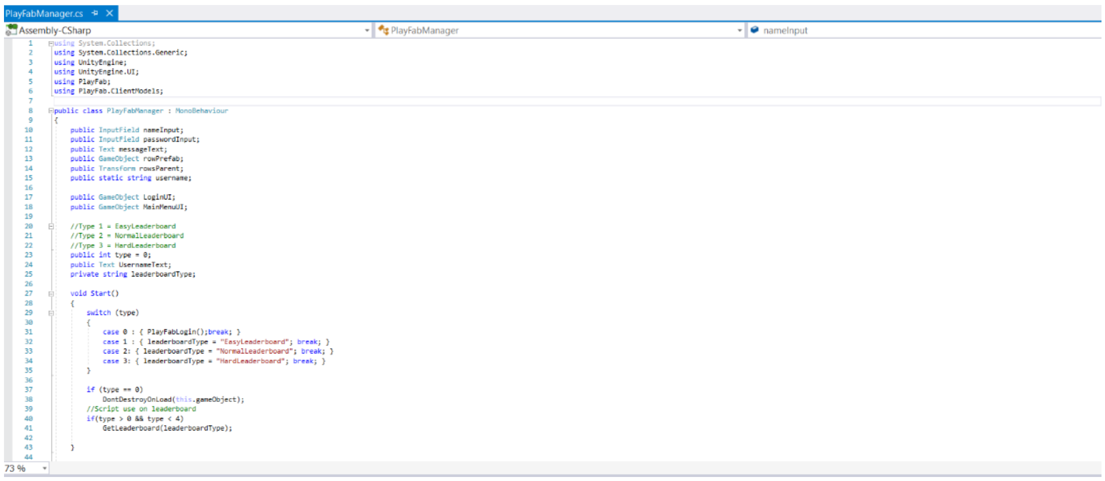
Main Menu (Script)

Script	MainMenu
Main Menu UI	MainMenuUI
Choose Stage UI	ChooseStageUI
Easy Leaderboard UI	EasyLeaderboardUI
Normal Leaderboard UI	NormalLeaderboardUI
Hard Leaderboard UI	HardLeaderboardUI
Login UI	LoginUI

Figure 5.3: Inspector panel of "Main Menu Manager" script in Unity

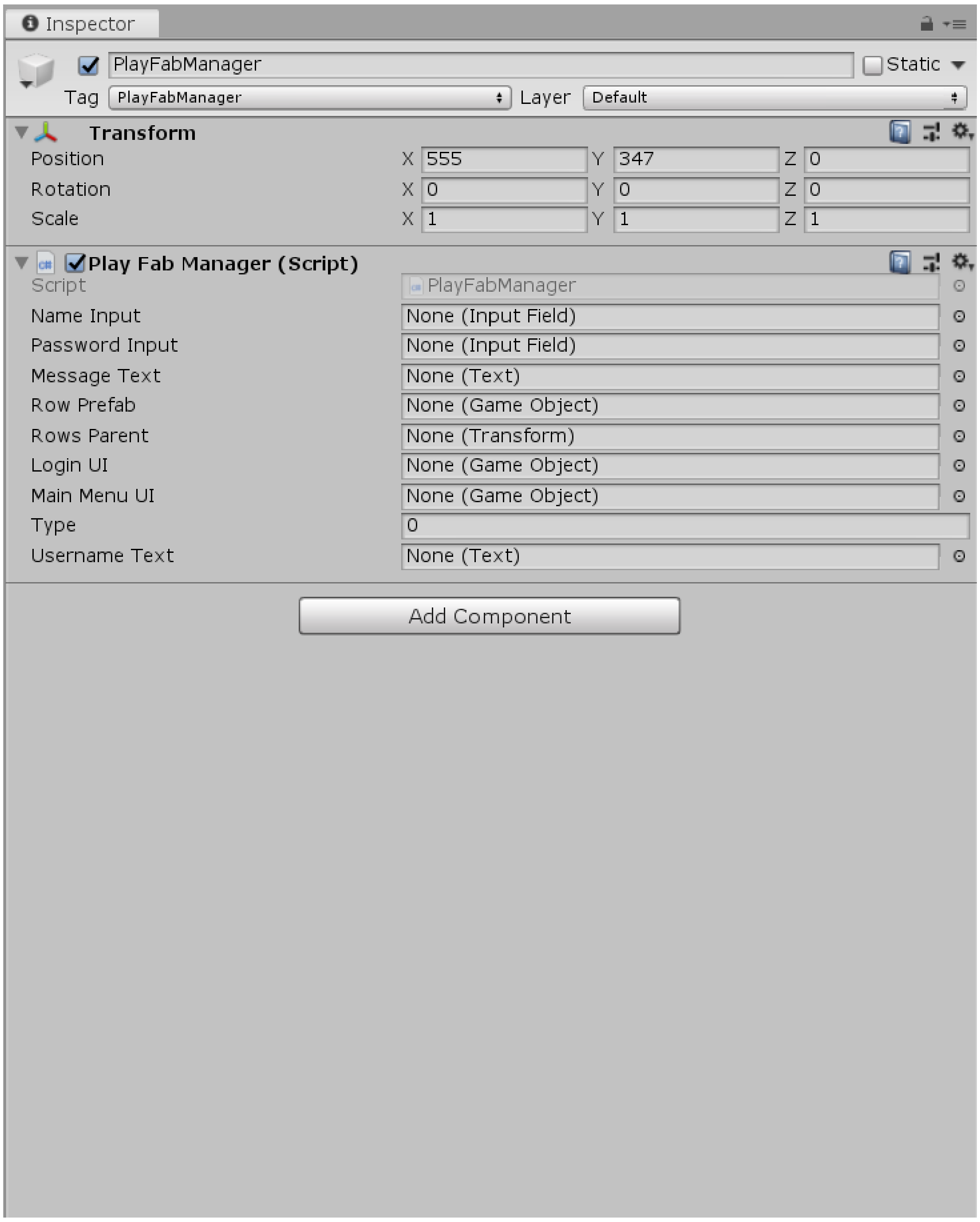
5.2.2 PlayFab Manager

The “PlayFab Manager” script is used to setup the connection between PlayFab server and the system. This connection provides the function for the system to get and send user data which is the username and password to the PlayFab server in order to authenticate the user during the login or register process. This script also needed for the "Leaderboard User Interface" since every record for the FPS game will also save in the PlayFab server. The GameObject containing this script will be called in Unity using DontDestroyOnload() function so that it can still be able to exist after the scene have changed from "Main Menu" scene to the "Stage" scene. Figure 5.4 shows the coding of "Playfab Manager" while Figure 5.5 is the inspector panel of "Playfab Manager" script in Unity.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using PlayFab;
using PlayFab.ClientModels;

public class PlayFabManager : MonoBehaviour
{
    public InputField nameInput;
    public InputField passwordInput;
    public Text messageText;
    public GameObject rowPrefab;
    public Transform rowsParent;
    public static string username;

    public GameObject LoginUI;
    public GameObject MainMenuUI;

    //Type 1 = EasyLeaderboard
    //Type 2 = NormalLeaderboard
    //Type 3 = HardLeaderboard
    public int type = 0;
    public Text UsernameText;
    private string leaderboardType;

    void Start()
    {
        switch (type)
        {
            case 0 : { PlayFabLogin();break; }
            case 1 : { leaderboardType = "EasyLeaderboard"; break; }
            case 2: { leaderboardType = "NormalLeaderboard"; break; }
            case 3: { leaderboardType = "HardLeaderboard"; break; }
        }

        if (type == 0)
            DontDestroyOnLoad(this.gameObject);
        //Script use on leaderboard
        if(type > 0 && type < 4)
            GetLeaderboard(leaderboardType);

    }

```

Figure 5.4: Coding of "Playfab Manager" script


Inspector

PlayFabManager Static

Tag PlayFabManager Layer Default

Transform

Position	X 555	Y 347	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Play Fab Manager (Script)

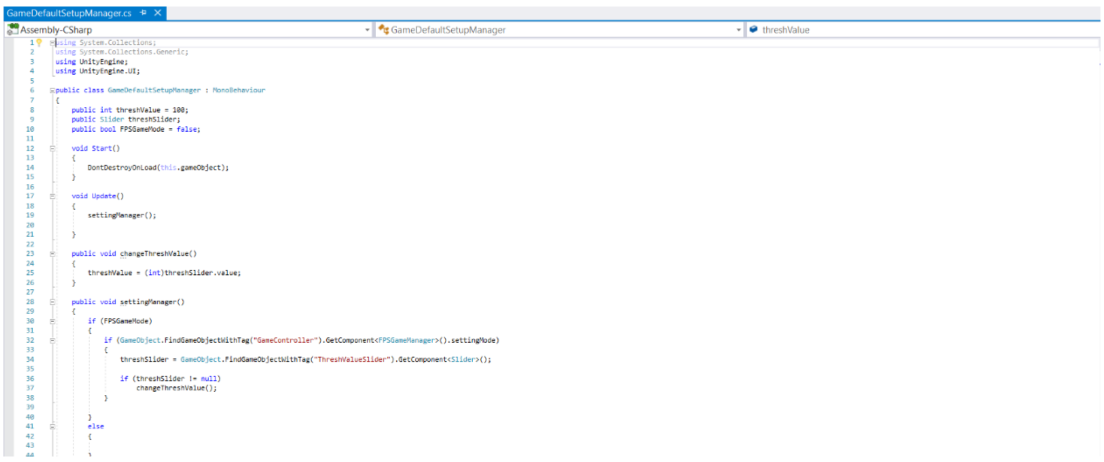
Script	PlayFabManager
Name Input	None (Input Field)
Password Input	None (Input Field)
Message Text	None (Text)
Row Prefab	None (Game Object)
Rows Parent	None (Transform)
Login UI	None (Game Object)
Main Menu UI	None (Game Object)
Type	0
Username Text	None (Text)

Add Component

Figure 5.5: Inspector panel of "Playfab Manager" script in Unity

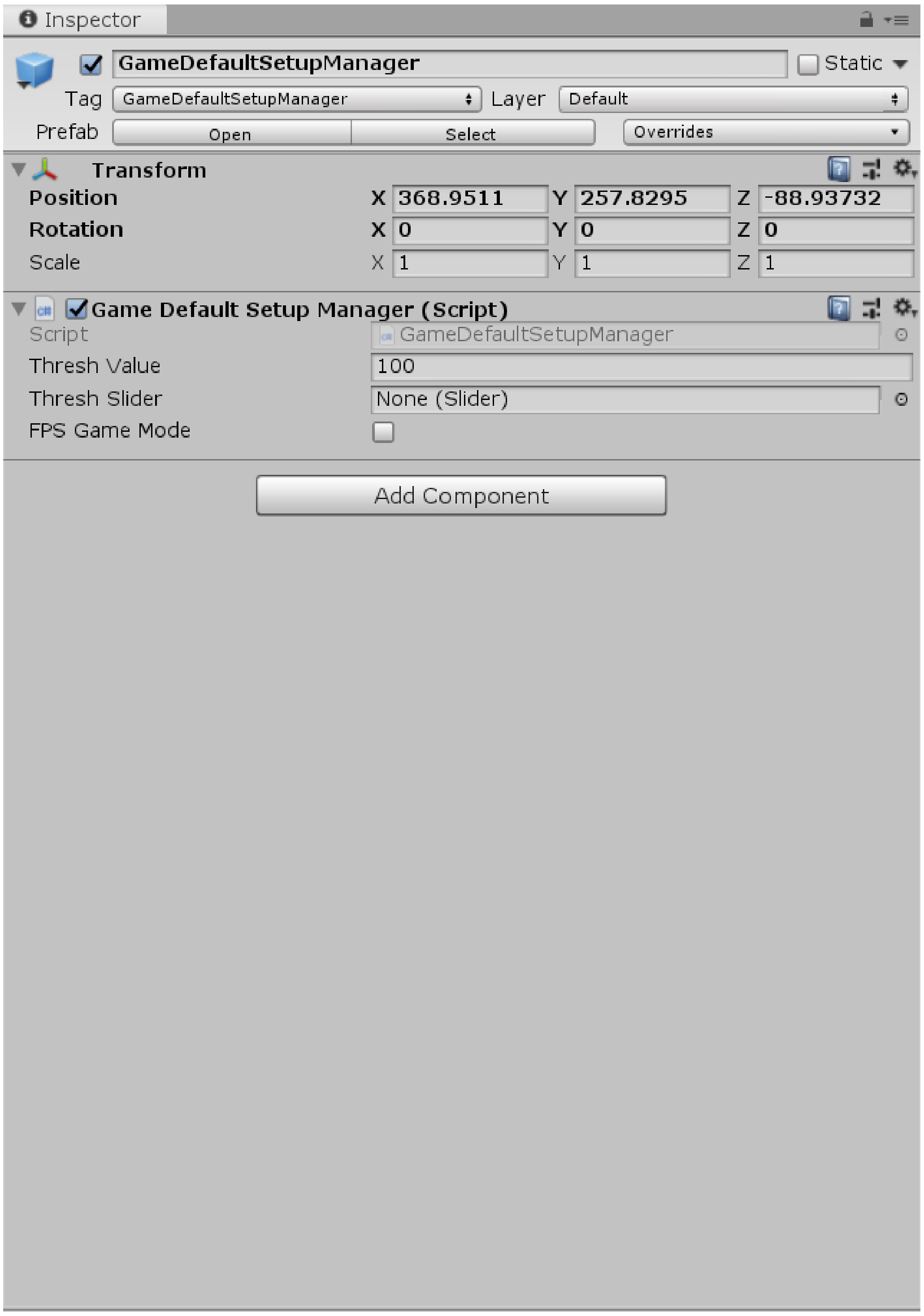
5.2.3 Game Default Setup Manager

The “Game Default Setup Manager” save the variable which will be used throughout the system which is the “thresh” value. the “thresh” value is the constant that will be use by the image detection script during the FPS game and it can be change by the user in “Setting User Interface”. By calling the DontDestroyOnload() function on the GameObject that this script attached on, the “thresh” value will maintain the value that the user has changed to ensure the image detection process works fine for the user’s device. Figure 5.6 is the C# coding in “Game Default Setup Manager” script while Figure 5.7 is the interface of “Game Default Setup Manager” in the inspector panel of Unity.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class GameDefaultSetupManager : MonoBehaviour
{
    public int threshValue = 100;
    public Slider threshSlider;
    public bool FPSGameMode = false;

    void Start()
    {
        DontDestroyOnLoad(this.gameObject);
    }

    void Update()
    {
        settingManager();

    }

    public void changeThreshValue()
    {
        threshValue = (int)threshSlider.value;
    }

    public void settingManager()
    {
        if (FPSGameMode)
        {
            if (GameObject.FindGameObjectWithTag("GameController").GetComponent<FPSGameManager>().settingMode)
            {
                threshSlider = GameObject.FindGameObjectWithTag("ThreshValueSlider").GetComponent<Slider>();

                if (threshSlider != null)
                    changeThreshValue();
            }

        }
        else
        {

        }
```

Figure 5.6: Coding of "Game Default Setup Manager" script



Inspector

GameDefaultSetupManager Static

Tag GameDefaultSetupManager Layer Default

Prefab Open Select Overrides

Transform

Position X 368.9511 Y 257.8295 Z -88.93732

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Game Default Setup Manager (Script)

Script GameDefaultSetupManager

Thresh Value 100

Thresh Slider None (Slider)

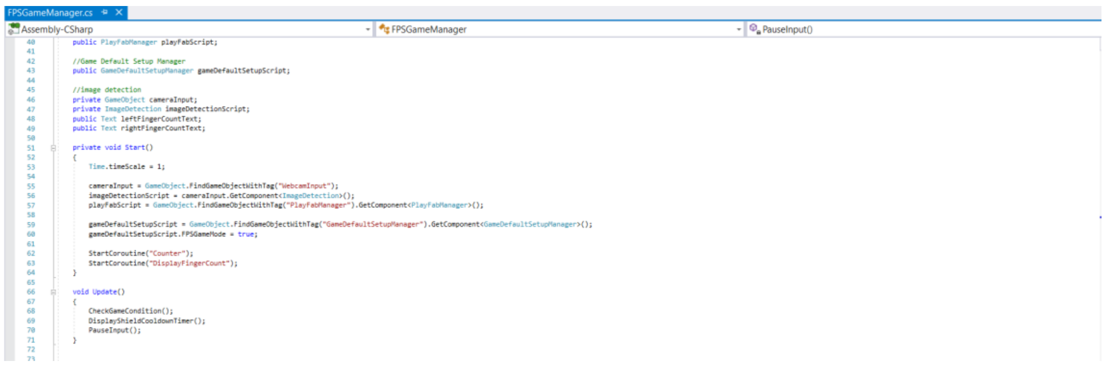
FPS Game Mode ☐

Add Component

Figure 5.7: Inspector panel of "Game Default Setup Manager" script in Unity

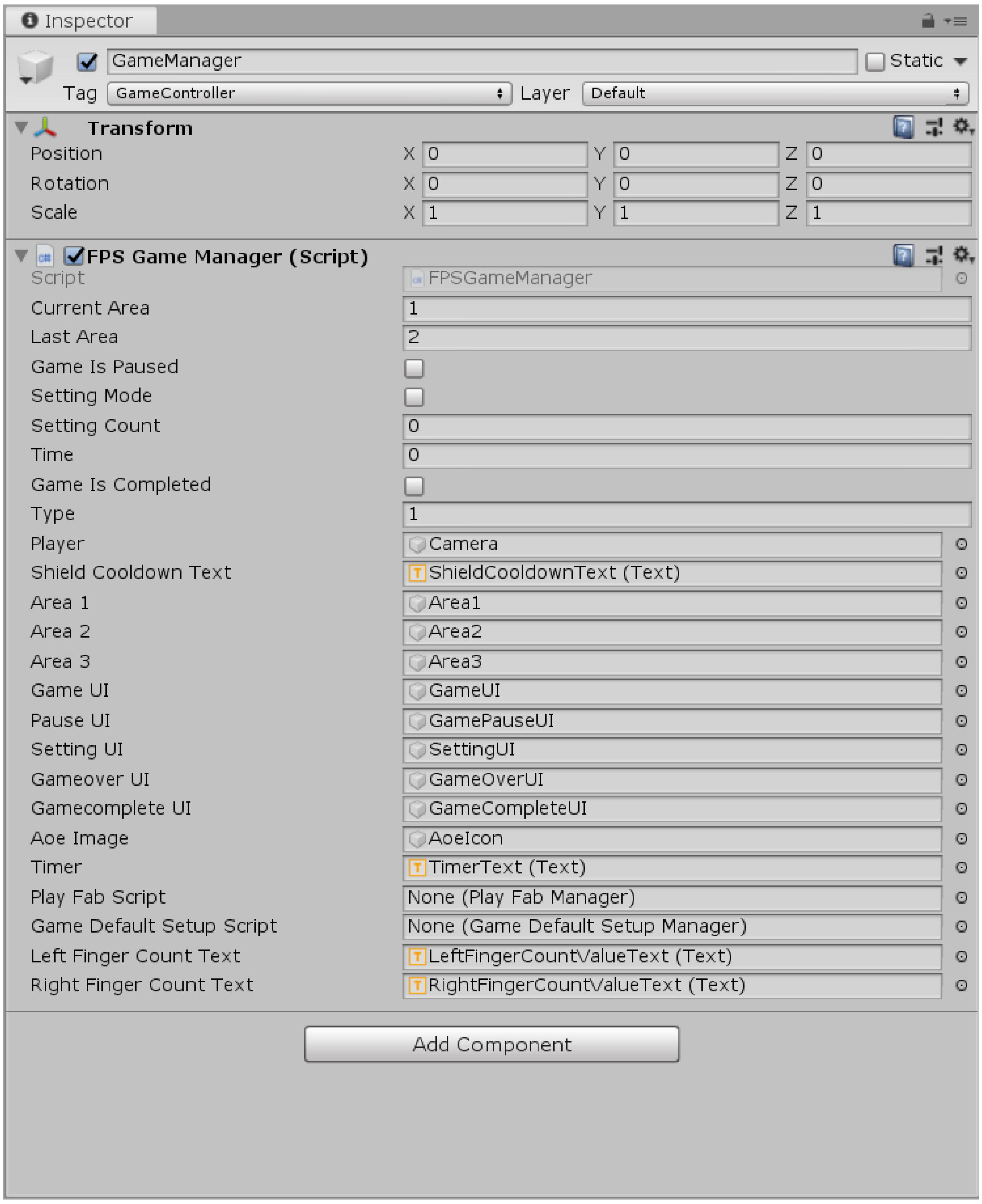
5.2.4 FPS Game Manager

"FPS Game Manager" is the C# script used to determine the current progress of the FPS game. This script will continue tracing the condition of the game which includes the time used, player health point and the drone amount currently existed in the scene. "FPS Game Manager" script will transfer the player in the game to the next area when the drone in the previous area has confirmed to be terminated. When the player decides to pause the game, "FPS Game Manager" script will determine this instruction and temporary stop the time count throughout the gaming process which eventually freeze the movement and behavior of the drone in the FPS game. Figure 5.8 shows the coding in "FPS Game Manager" script and Figure 5.9 shows the interface value of "FPS Game Manager" script inside inspector panel in Unity.



```
    public PlayFabManager playFabScript;

    //Game Default Setup Manager
    public GameDefaultSetupManager gameDefaultSetupScript;

    //image detection
    private GameObject cameraInput;
    private ImageDetection imageDetectionScript;
    public Text leftFingerCountText;
    public Text rightFingerCountText;

    private void Start()
    {
        Time.timeScale = 1;

        cameraInput = GameObject.FindGameObjectWithTag("WebcamInput");
        imageDetectionScript = cameraInput.GetComponent<ImageDetection>();
        playFabScript = GameObject.FindGameObjectWithTag("PlayFabManager").GetComponent<PlayFabManager>();

        gameDefaultSetupScript = GameObject.FindGameObjectWithTag("GameDefaultSetupManager").GetComponent<GameDefaultSetupManager>();
        gameDefaultSetupScript.FPSGameMode = true;

        StartCoroutine("Counter");
        StartCoroutine("DisplayFingerCount");
    }

    void Update()
    {
        CheckGameCondition();
        DisplayShieldCooldownTimer();
        PauseInput();
    }
```

Figure 5.8: Coding of "FPS Game Manager" script



Inspector

GameManager ☐ Static

Tag GameController Layer Default

Transform

Position	X	0	Y	0	Z	0
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

FPS Game Manager (Script)

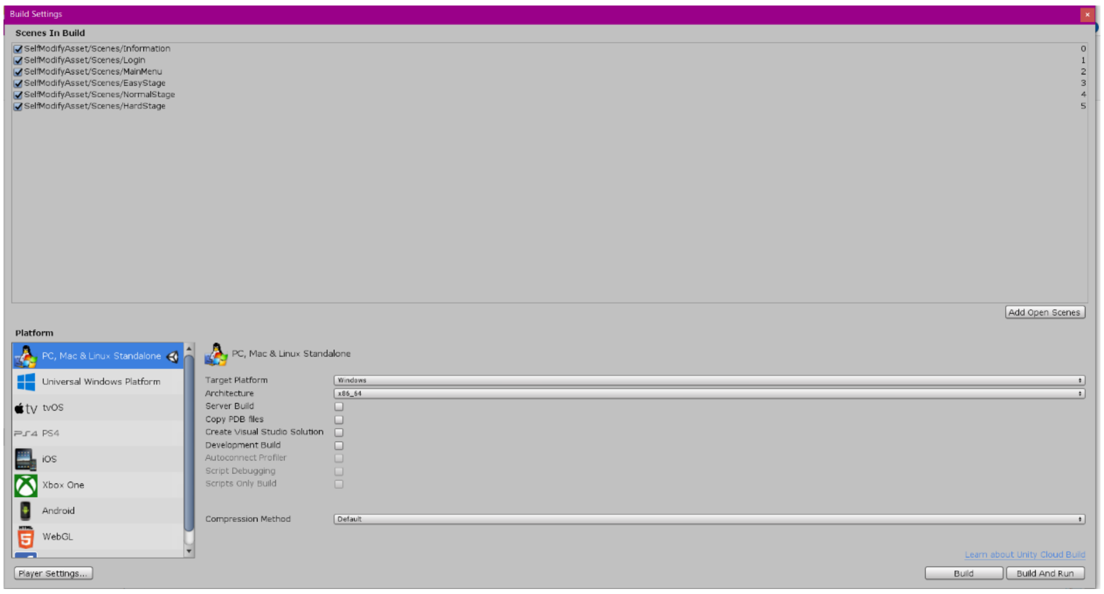
Script	FPSGameManager
Current Area	1
Last Area	2
Game Is Paused	☐
Setting Mode	☐
Setting Count	0
Time	0
Game Is Completed	☐
Type	1
Player	Camera
Shield Cooldown Text	ShieldCooldownText (Text)
Area 1	Area1
Area 2	Area2
Area 3	Area3
Game UI	GameUI
Pause UI	GamePauseUI
Setting UI	SettingUI
Gameover UI	GameOverUI
Gamecomplete UI	GameCompleteUI
Aoe Image	AoeIcon
Timer	TimerText (Text)
Play Fab Script	None (Play Fab Manager)
Game Default Setup Script	None (Game Default Setup Manager)
Left Finger Count Text	LeftFingerCountValueText (Text)
Right Finger Count Text	RightFingerCountValueText (Text)

Add Component

Figure 5.9: Inspector panel of "FPS Game Manager" script in Unity

5.3 System results

The completed system is able to build using Unity3D as shown in Figure 5.10. It can be run mainly under the computer with Windows operating system. Figure 5.11 is the folder which has been created using Unity3D. The user can open the game by selecting the application file with name "PSM2 Hand Gesture Control FPS Game.exe".



Build Settings

Scenes In Build

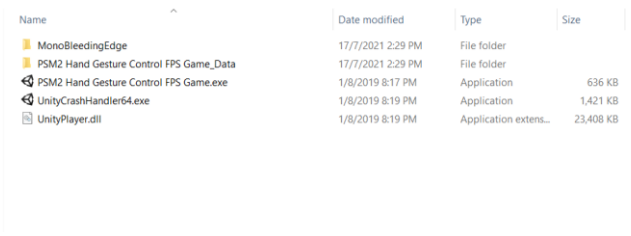
☑ SelfModifyAsset/Scenes/Information	0
☑ SelfModifyAsset/Scenes/Login	1
☑ SelfModifyAsset/Scenes/MainMenu	2
☑ SelfModifyAsset/Scenes/EasyStage	3
☑ SelfModifyAsset/Scenes/NormalStage	4
☑ SelfModifyAsset/Scenes/HardStage	5

Platform: PC, Mac & Linux Standalone

Target Platform	Windows
Architecture	x86_64
Server Build	☐
Copy PDB files	☐
Create Visual Studio Solution	☐
Development Build	☐
Autoconnect Profiler	☐
Script Debugging	☐
Scripts Only Build	☐
Compression Method	Default

Add Open Scenes
PC, Mac & Linux Standalone
Universal Windows Platform
tvOS
PS4
iOS
Xbox One
Android
WebGL
Player Settings...
Learn about Unity Cloud Build
Build
Build And Run

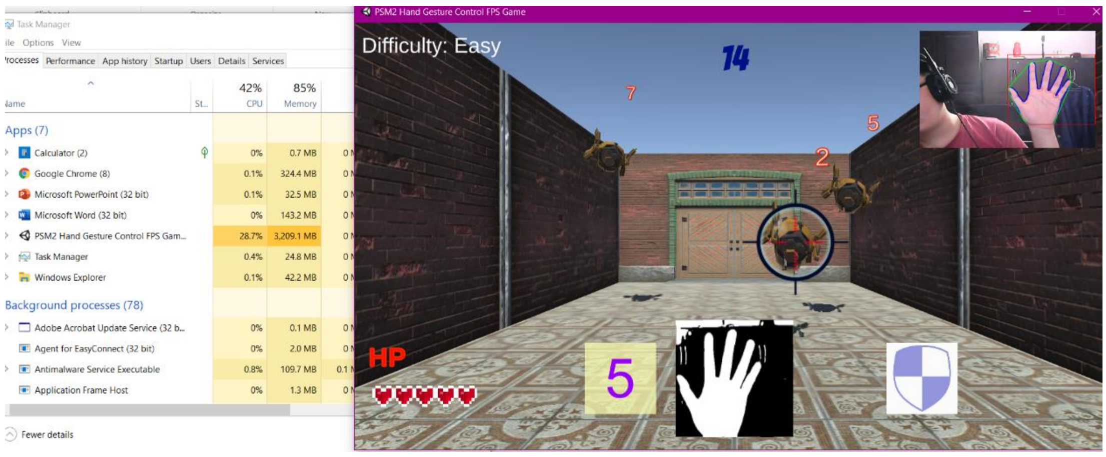
Figure 5.10: System building using Unity3D



Name	Date modified	Type	Size
MonoBleedingEdge	17/7/2021 2:29 PM	File folder	
PSM2 Hand Gesture Control FPS Game_Data	17/7/2021 2:29 PM	File folder	
PSM2 Hand Gesture Control FPS Game.exe	1/8/2019 8:17 PM	Application	636 KB
UnityCrashHandler64.exe	1/8/2019 8:19 PM	Application	1,421 KB
UnityPlayer.dll	1/8/2019 8:19 PM	Application extens...	23,408 KB

Figure 5.11: Contents of the Completed System

The application does consume a lot of RAM memory and CPU processing time as show in Figure 5.12 mainly due to the image processing function. Some delay may be happened during the gameplay after some time.


Task Manager

File Options View

Processes | Performance | App history | Startup | Users | Details | Services

Name	St...	42% CPU	85% Memory	
Apps (7)				
Calculator (2)		0%	0.7 MB	0
Google Chrome (8)		0.1%	324.4 MB	0
Microsoft PowerPoint (32 bit)		0.1%	32.5 MB	0
Microsoft Word (32 bit)		0%	143.2 MB	0
PSM2 Hand Gesture Control FPS Gam...		28.7%	3,209.1 MB	0
Task Manager		0.4%	24.8 MB	0
Windows Explorer		0.1%	42.2 MB	0
Background processes (78)				
Adobe Acrobat Update Service (32 b...		0%	0.1 MB	0
Agent for EasyConnect (32 bit)		0%	2.0 MB	0
Antimalware Service Executable		0.8%	109.7 MB	0.1
Application Frame Host		0%	1.3 MB	0

Fewer details

Figure 5.12: Usage of CPU and RAM Memory of the Completed System

5.4 Testing

Black-box testing will use as the testing method for this project. The application will be delivered to the tester and the tester will try to run the application using their own personal computer or laptop and webcam. After the tester has complete the testing, a Google form will use as the questionnaire to retrieve the testing result of the tester to use as the material on determine the usability and accuracy of the system. Table 5-1 shows the question in the Google form. The score for each value will scale from 1 to 5 with strongly disagree as 1 while strongly agree as 5.

Table 5-1: Question in the Google form

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
Specification					
RAM memory					
Webcam pixel					
Satisfaction Level					

The user interface is well design.					
The instruction in the game is clear.					
The game difficulty is well defined.					
The game is interesting to play.					
The game can be played smoothly.					
The accuracy of the image detection is high.					
The design of the environment is good.					
Input Accuracy					
Fire Command (2 Fingers and Left Click)					
Switch Target Command (3 Fingers and Left Click)					
Shield Command (5 Fingers and Left Click)					

After the system has been tested by 5 testers, all the responses have been gathered and concluded in Table 5-2. All of the testers have the computer with RAM memory that is equal or greater than 8 Gigabytes, which is a common specification for recent computers. However, the webcam pixel for all the testers has a big range of differences from 0.307 to 2.072 megapixel. After calculating the average value for the questions, it can be found that there are 2 main criteria which may not fulfil the expectation of the tester which is the question "The game can be played smoothly" with average score 2.8 and "The accuracy of image detection is high" with average score 2.2. However, the input accuracy for all 3 commands has a score which is higher than the average. This does prove that even the accuracy of image detection may not satisfy the user, but the system is able to understand the command when at some instant during the image detection process.

Table 5-2: Response from the Tester

Question	Tester 1	Tester 2	Tester 3	Tester 4	Tester 5	Average
Specification						
RAM memory (GB)	8	8	8	8	12	-
Webcam pixel (Megapixel)	0.307	1.0	0.307	0.307	2.073	-
Satisfaction Level						
The user interface is well design.	3	4	4	3	4	3.6
The instruction in the game is clear.	4	4	5	3	3	3.8
The game difficulty is well defined.	3	4	4	3	4	3.6
The game is interesting to play.	3	4	3	3	4	3.4
The game can be played smoothly.	3	2	3	3	3	2.8
The accuracy of the image detection is high.	3	2	1	2	3	2.2
The design of the environment is good.	4	4	3	2	5	3.6
Input Accuracy						
Fire Command (2 Fingers and Left Click)	4	3	5	2	4	3.6
Switch Target Command (3 Fingers and Left Click)	4	3	4	2	3	3.2
Shield Command (5 Fingers and Left Click)	4	3	4	4	2	3.4

It can be concluded that the completed system still needs some optimization for the image processing algorithm to ensure the detection process can be more accurate and less burden to the CPU and GPU of the computer.

5.5 Chapter Summary

Subchapter 5.2 describe about the coding of system's main function. All of the function will be written in C# script and attached to the game object inside Unity3D.

Subchapter 5.3 describes about the results of the completed system. The system is built mainly for Windows operating system and will be able to run as an application.

Subchapter 5.4 discuss about the result of the testing. According to the result, the main weakness of the completed system is the accuracy of the image detection and the game performance.

CHAPTER 6

CONCLUSION

6.1 Introduction

The completed system does provide the function for the user to control the player in the FPS game with 3 types of hand gesture. The system does also provide some gamification features using user login and leaderboard and a tutorial for the player to go through and understand about the game play of the FPS game.

6.2 Achievement of Project Objectives

The requirement of first-person shooter games has been determined from this project which is the accuracy of hand gestures under real-time conditions. An auto aiming function has been provided in the system to reduce the need for the user to change target using any hand gesture in order to ensure the gameplay can focus on the shape of the hand but not the movement of the hand. The FPS game is able to run under any personal computer or laptop with the use of a webcam.

The image processing algorithm that has been used in the system is finding the convexity defects and determining the numbers of fingers in order to understand which kind of hand gesture is it. This will not limit the type of hand gesture for the input since its all depends on the numbers of fingers but does not have a strictly hand gesture.

According to the evaluation on the completed system, the possibility of using a webcam to replace other hand gestures is still quite low since it still depends a lot on the specification of the user's webcam and the environment of the user. However, it does prove that the game is still able to detect hand gestures from webcam successfully but still needs more improvement on the algorithm used.

6.3 Suggestions for Future Improvement

According to the feedback of the tester, it can conclude that the image detection process is quite memory consuming and causes the game hard to run smoothly. The memory optimization still needs to be further improved to increase the satisfaction level of the user.

Currently the completed system only can detect the hand gesture with real-time input which has the weak point that the image detection background needs to be in dark color to increase the accuracy of the hand gesture detection. For future improvement, it is possible that the system can be trained using a neural network to have the ability to detect hand position under complicated backgrounds. It is quite time consuming but can surely make the image detection process smoother and more accurate.

REFERENCES

Cardamone, L., Yannakakis, G. N., Togelius, J., & Lanzi, P. L. (2011, April). Evolving interesting maps for a first person shooter. In *European Conference on the Applications of Evolutionary Computation* (pp. 63-72). Springer, Berlin, Heidelberg.

Chen, W. H., Lin, Y. H., & Yang, S. J. (2010, June). A generic framework for the design of visual-based gesture control interface. In 2010 5th IEEE Conference on Industrial Electronics and Applications (pp. 1522-1525). IEEE.

Chen, Z. H., Kim, J. T., Liang, J., Zhang, J., & Yuan, Y. B. (2014). Real-time hand gesture recognition using finger segmentation. *The Scientific World Journal, 2014.*

Colzato, L. S., Van Leeuwen, P. J., Van Den Wildenberg, W., & Hommel, B. (2010). DOOM'd to switch: superior cognitive flexibility in players of first person shooter games. Frontiers in psychology, 1, 8.

Dardas, N. H., Silva, J. M., & El Saddik, A. (2014). Target-shooting exergame with a hand gesture control. Multimedia tools and applications, 70(3), 2211-2233.

Denisova, A., & Cairns, P. (2015, April). First person vs. third person perspective in digital games: do player preferences affect immersion?. In Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (pp. 145-148).

Grimshaw, M., Charlton, J., & Jagger, R. (2011). First-person shooters: Immersion and attention.Eludamos. Journal for Computer Game Culture, 5(1), 29-44.

Haas, J. K. (2014). A history of the unity game engine.

Hamza, A., Anand, R., Shivhare, P., & Gaurav, A. (2017). Hand Gesture Recognition Applications. International Journal of Interdisciplinary Research, 13, 2073-2075.

Hitchens, M. (2011). A Survey of First-person Shooters and their Avatars. Game Studies, 11(3), 96-120.

Kaehler, A., & Bradski, G. (2016). *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. " O'Reilly Media, Inc.".

Kuikkaniemi, K., Laitinen, T., Turpeinen, M., Saari, T., Kosunen, I., & Ravaja, N. (2010, April). The influence of implicit and explicit biofeedback in first-person shooter games. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 859-868).

Ma'asum, F. F. M., Sulaiman, S., & Saparon, A. (2015). An overview of hand gestures recognition system techniques. In *IOP Conference Series: Materials Science and Engineering* (Vol. 99, No. 1, p. 012012). IOP Publishing.

Roccetti, M., Marfia, G., & Semeraro, A. (2012). Playing into the wild: A gesture-based interface for gaming in public spaces. Journal of Visual Communication and Image Representation, 23(3), 426-440.

Weber, R., Behr, K. M., Tamborini, R., Ritterfeld, U., & Mathiak, K. (2009). What do we really know about first-person-shooter games? An event-related, high-resolution content analysis. *Journal of Computer-Mediated Communication*, *14*(4), 1016-1037.

Therrien, C. (2015). Inspecting video game historiography through critical lens: Etymology of the first-person shooter genre. *Game Studies*, *15*(2).

Zhou, Q., Miller, C. J., & Bassilious, V. (2008, May). First person shooter multiplayer game traffic analysis. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)* (pp. 195-200). IEEE.

Appendix A