

GAMIFICATION OF LEARNING OBESITY

BY

PANG QING SEN

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2024

REPORT STATUS DECLARATION FORM

Title: _____ Gamification in Learning Obesity _____

Academic Session: _____ June 2024 _____

I _____ PANG QING SEN _____
(CAPITAL LETTER)

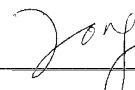
declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

NO 12 JLN HARMONIUM____
23/6 TAMAN DESA TEBRAU
81100 JOHOR BAHRU _____

Tong Dong Ling

Supervisor's name

Date: _____ 12/9/2024 _____

Date: _____ 12 Sep 2024 _____

Universiti Tunku Abdul Rahman			
Form Title : Sample of Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY/INSTITUTE* OF ___Information and Communication Technology ___
UNIVERSITI TUNKU ABDUL RAHMAN

Date: ___11 Sep 2024_____

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that ___Pang Qing Sen (ID No: 21ACB06605) has completed this final year project/ dissertation/ thesis* entitled “Gamification of Learning Obesity” under the supervision of ___Ts Tong Dong Ling_____ (Supervisor) from the Department of ___Computer Science___, Faculty/Institute* of ___Information and Communication Technology_____, and ___n/a_____ (Co-Supervisor)* from the Department of ___n/a_____, Faculty/Institute* of ___n/a_____.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,




(PANG QING SEN)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled “**GAMIFICATION OF LEARNING OBESITY**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :  _____

Name : _____ Pang Qing Sen _____

Date : _____ 12/09/2024 _____

ACKNOWLEDGEMENTS

I would like to thank to my supervisor, Ts Dr Tong Dong Ling, for guiding me along this game development project. She provided me with a lot of suggestions and listened to my opinion about the game development, and I am deeply grateful for your guidance and support throughout this journey. Your mentorship has been invaluable, and I am truly thankful for the knowledge and insights you have shared with me.

I would also like to express my profound appreciation to my parents and family for their unwavering love, support, and encouragement throughout this project journey. Their constant belief in my abilities has been a source of strength and motivation, and I am grateful for their presence in my life.

ABSTRACT

Obesity is a widespread problem worldwide, the issue is rapidly increasing in more and more countries, including both developed and developing nations, such as China, the United States, and Malaysia. It is essential to educate people about obesity early to prevent its progression and development of the obesity. Gamification is a powerful tool that can help people in this situation as it encourages to learn new things in an entertaining and exciting way. The central tenet of the project is gamification's use in teaching players about the impact of obesity on their characters. The project will start with a thorough review of the current state of gamification in broad areas and a search for the most appropriate ways to transfer the necessary game elements to our product. The game elements that attract players do not necessarily convey knowledge as well. This ensures that the selected game elements should be not only suitable for attracting players but also effective in knowledge implementation. After the extensive analysis and design phases, the next stage of the project involves the implementation, where our ideas are put into reality through coding and subsequent testing. Once the game has been developed, tested, and validated, it will be deployed to Windows operating system, where people can obtain it for their own use.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	x
LIST OF TABLES	xii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1 INTRODUCTION	1
1.1 Problem Statement and Motivation	1
1.2 Objectives	1
1.3 Project Scope and Direction	2
1.4 Contributions	2
1.5 Report Organization	2
CHAPTER 2 LITERATURE REVIEW	4
2.1 EzBlock Studio	4
2.2 IoT in a Box	10
2.3 Skydrop Smart Sprinkler Controller	12
2.4 IoTool	14
2.5 RaspController	16
CHAPTER 3 SYSTEM METHODOLOGY/APPROACH (FOR DEVELOPMENT-BASED PROJECT)	17
CHAPTER 3 SYSTEM MODEL (FOR RESEARCH-BASED PROJECT)	17
3.1 System Design Diagram/Equation	17

3.1.1	System Architecture Diagram	17
3.1.2	Use Case Diagram and Description	17
3.1.3	Activity Diagram	18
CHAPTER 4	SYSTEM DESIGN	19
4.1	System Block Diagram	19
4.2	System Components Specifications	20
4.3	Circuits and Components Design	21
4.4	System Components Interaction Operations	22
CHAPTER 5	SYSTEM IMPLEMENTATION (FOR DEVELOPMENT- BASED PROJECT)	25
CHAPTER 5	EXPERIMENT/SIMULATION (FOR RESEARCH- BASED PROJECT)	25
5.1	Hardware Setup	25
5.2	Software Setup	30
5.3	Setting and Configuration	41
5.4	System Operation (with Screenshot)	52
5.5	Implementation Issues and Challenges	62
5.6	Concluding Remark	66
CHAPTER 6	SYSTEM EVALUATION AND DISCUSSION	70
6.1	System Testing and Performance Metrics	70
6.2	Testing Setup and Result	75
6.3	Project Challenges	82
6.4	Objectives Evaluation	89
6.5	Concluding Remark	91

CHAPTER 7 CONCLUSION AND RECOMMENDATION	101
7.1 Conclusion	101
7.2 Recommendation	103
REFERENCES	109
APPENDIX	112
WEEKLY LOG	145
POSTER	150
PLAGIARISM CHECK RESULT	151
FYP2 CHECKLIST	153

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.1	The RGB stream, 3D points and final mesh by Johnso et.al. [8]	10
Figure 3.2.1	System Methodology Overview	18
Figure 3.2.2	Flow Chart of the system	21
Figure 3.2.3	Assets found from itch.io	21
Figure 3.2.4	Assets found from Unity Asset Store	22
Figure 3.2.5	Assets creation with Paint	23
Figure 3.2.6	Unity Hub for unity projects management	23
Figure 3.2.7	Unity Editor for game development	24
Figure 3.2.8	Visual Studio for C# programming	24
Figure 3.3	Timeline for Project II	26
Figure 4.1.1	System Block Diagram	27
Figure 4.1.2	Activity Diagram of the system	28
Figure 4.1.3	System Hierarchy Diagram	28
Figure 4.2.1	Player Game Object Animation	34
Figure 4.2.2	Player Input Configuration Settings	35
Figure 4.2.3	Box Collider 2D and Rigid Body 2D for Player Game Object	35
Figure 4.2.4	The Weapons and The Player Game Object	36
Figure 4.2.5	The Scripts attached to the Player Game Object	36
Figure 4.2.6	The Animation for Enemies and Victims Objects	37
Figure 4.2.7	The Game Objects that Used the Collider 2D	38
Figure 4.2.8	The Scripts Used by The Enemy and Victim Game Objects	38
Figure 4.2.9	Map for First Level	39
Figure 4.2.10	Map for Second Level	39
Figure 4.2.11	Map for Third Level	40
Figure 4.2.12	Map for Level 4	40
Figure 4.2.13	Map for Level 5	41

Figure 4.2.14	Buttons used in the game	45
Figure 4.2.15	Menus in this game	45
Figure 4.2.16	Scoreboards in this game	46
Figure 4.2.17	Example of pages design in Level 1	46
Figure 4.2.18	Dialogue box and Message Box in the game	47
Figure 4.3.1	The structure of the game manager scene and main menu scene	49
Figure 5.2.1	The login page of the Unity ID	57
Figure 5.2.2	The registration page of the Unity ID	57
Figure 5.2.3	The downloaded executable file	58
Figure 5.2.4	The Unity Hub Installation Process	58
Figure 5.2.5	The Unity Hub Login Page	59
Figure 5.2.6	The Unity Hub Page After Successful Logged In	59
Figure 5.2.7	The Installs Page in Unity Hub	60
Figure 5.2.8	The Visual Studio Download Page	61
Figure 5.2.9	The Downloaded Executable File and the Visual Studio Installation Process	61
Figure 5.2.10	The Installed Version of the Visual Studio	62
Figure 5.3.1	The Interface of Unity Hub	63
Figure 5.3.2	The new project creation and loading screen	64
Figure 5.3.3	The Interface of Unity Editor	65
Figure 5.3.4	Create new file in Unity Editor	65
Figure 5.3.5	Create New Game Object and Add New Components in Unity Editor	66
Figure 5.3.6	Run the game in development mode	67
Figure 5.3.7	Build the game and export to executable file	68
Figure 6.2.1	Development build and auto-connect profiler	99
Figure 6.2.2	Open the profiler in Unity Editor	100
Figure 6.2.3	Profiler in Unity Editor	100
Figure 6.2.4	Build settings for WebGL	102

LIST OF TABLES

Table Number	Title	Page
Table 2.1	Comparison Between Gamification in Different Fields	12
Table 2.2	Comparison Between Effect of Obesity toward Human Organs	13
Table 3.1.1	Specifications of Laptop	16
Table 3.1.2	Software and Programming Environment	17
Table 4.2.1	Ranking Classification Based on Score Range	48
Table 5.1	Specifications of Laptop	56
Table 5.4.1	System Operation for the Main Menu and Tutorial	69
Table 5.4.2	System Operation for Level 1	70
Table 5.4.3	System Operation for Level 2	73
Table 5.4.4	System Operation for Level 3	76
Table 5.4.5	System Operation for Level 4	79
Table 5.4.6	System Operation for Level 5	82
Table 5.4.7	System Operation for Ending	85
Table 6.2.1	Result of Functional Testing for Main menu and Tutorial	91
Table 6.2.2	Result of Functional Testing for Level 1	91
Table 6.2.3	Result of Functional Testing for Level 2	93
Table 6.2.4	Result of Functional Testing for Level 3	94
Table 6.2.5	Result of Functional Testing for Level 4	95
Table 6.2.6	Result of Functional Testing for Level 5	97
Table 6.2.7	Result of Functional Testing for Ending	98
Table 6.2.8	Edge Case Testing Result	99
Table 6.2.9	Non-functional Testing Result	101

LIST OF ABBREVIATIONS

<i>AI</i>	Artificial Intelligence
<i>CPU</i>	Central Processing Unit
<i>IDE</i>	Integrated Development Environment
<i>FPS</i>	Frate Per Second
<i>MCO</i>	Movement Control Orders
<i>IGER</i>	Intelligent Gaming Engine for Rehabilitation
<i>SVR</i>	Peripheral Vascular Resistance
<i>RPG</i>	Role Play Game
<i>OOP</i>	Object-Oriented Programming
<i>OS</i>	Operating System
<i>UI</i>	User Interface

Chapter 1

Introduction

In this chapter, we will discuss on the problem statement, motivation, objectives and scope of the project and its direction, as well as the contribution of this project.

1.1 Problem Statement and Motivation

Obesity is one of the most serious issues that occurs in many countries in the world including Malaysia. Malaysia has confronted with the challenges regarding obesity and overweight populations. Based on the statistic reported by Tat et.al. [1], significant portions of the Malaysian population which is around 19.9% of them have suffered from obesity in 2019. The situation become even worse after many measurements that triggered by the COVID-19 pandemic such as the lockdowns and implementation of Movement Control Orders (MCO) by the Malaysian government. As reported by Chin et.al. [2], these enforcement and implementation of measurements then lead to changes in people's normal behavior such as spending lesser time on the outdoors activities and become less active day-to-day. Malaysian Health Ministry has taken action to address this issue to prevent it become more serious from time to time, by putting efforts to educate the public about the risks linked to obesity.

Obesity is not only famous as a direct cause of mortality, but it also recognized as a significant contributor to various health issues such as heart attacks, high blood pressure, elevated cholesterol levels, and strokes. Due to that reason, it is important to understand how obesity impacts different organs in the body, especially for those suffered from obesity. As an inspiration of that, we could implement a game that educate people regarding the negative effects of the obesity on different part of organs effectively, which is another way around the

traditional method such as a class lecture or an educational campaign. Therefore, this paper aims to demonstrate the practical implementation of game elements on education of the obesity, which will produce a functional game at the end of this project. Despite on assessing how effective gamification is in learning obesity; we also targeted to raise people's awareness about the effects of obesity and its associated health challenges.

This can be realized by implementing the game elements such as the scoring and reward system, a compelling storyline and immersive gameplay of the game. In the game, users will play as the main character and be actively involved in the activities that help body cells and other in-game characters that suffer from the effects of obesity, thus providing an in-depth exploration of the educational knowledge surrounding the effects of obesity on human organs. This approach lets users involve themselves with the subject matter at a microscopic level in the game and grants them a visually captivating and educational experience.

1.2 Objectives

In our paper, we proposed a functional game system that can utilize the gamification's key elements which can enhance the learning efficiency while motivating people to learn knowledge regarding the detrimental effects of obesity towards different human organs. Each of the objectives is outlined as follows:

- i) To investigate and identify suitable game elements conducive to gamified learning to enhance the efficiency of learning which makes the educational process more engaging through gaming.
- ii) To design and develop a 2D game that consist of 5 levels from the chosen game elements which the system should encourage individuals to learn about the negative effects of obesity on human organs through immersive gaming.
- iii) To elevate awareness of the public towards the effects of obesity and encourage them to take the action on maintaining a healthy lifestyle.

1.3 Project Scope and Direction

The scope of this project includes the identification and selection of the game attributes and elements in the early stage, which will guide by the principle that selected gamification elements should incorporate engaging elements to attract players which helps them learn the knowledge effectively while maintaining a simple but captivating game structure. The elements of the game consist of a simple storyline that can facilitate player engagement through narrative-driven gameplay and by RPG elements that can enhance the interactivity. In the subsequent stage of game design, this paper will adhere to educational principles albeit in a manner that avoids appearing overtly didactic or requiring players to acquire knowledge rigidly and statically. Instead, the game will clarify the knowledge throughout the gameplay experience. Moreover, user-friendliness is also important as players need only minimal prior

knowledge to play the game by using well-known or common controls like W, A, S, and D for character movement. Furthermore, the game will be developed in a 2D pixel-based format which the development process of the game could be simplify and ensure accessibility and playability for users.

In the system design phase, we will craft the game, design the levels, construct the story and determine the overall flow of the game. In this phase, we will also plan for the gameplay for each of the level, including how will the score being calculated, the knowledge delivered to the player, and design of each of the levels. After that, we continue to the transition into system implementation which turns our plans into a working system according to the constructed design. Subsequently, rigorous testing and detailed validation are performed to ensure the system works as expected. Finally, after successful testing of the system, the system is now ready for deployment on the operating system which allows users to use and play it directly on their OS. This structured approach provides a systematic and established overview of the project journey from the conversion of a conceptual system to a fully functional system.

1.4 Contributions

This paper conducts a thorough examination and implementation of the identified important game attributes and elements that helps for learning obesity. In this paper, we aim to validate the practicality of the proposed gamification concept which the design of the game is translated into a fully functional and executable game with the proposed methodology. This methodology includes the design of different components of the game such as the level design, scene design, game logic and the game story. For example, we create a storyline for each level, determining each determining each of the characters to be included in the game, planning the structure and design of each level, and ensuring a coherent and smooth flow for the game.

It is important to know that we perform the research on the actual development and implementation which will turn the conceptual ideas into a functional game since many of the research might not implement the conceptual ideas regarding the gamification of learning but limit themselves to the proposed study. In this paper, our approach involves not only the identification and selection of gamification elements but also their practical implementation in the creation of a functional game system that contributes to practical gamification implementation projects which uses some novel ideas to.

1.5 Report Organization

The structure of this report will be presented chapter by chapter as follows. Chapter 2 shows the papers reviewed of existing research on gamification concepts and their applications across various fields particularly in education, fitness, well-being and healthcare. In Chapter 3, we will propose the methodology for this project includes division of different phases of the project. As we move forward, Chapter 4 describes the system design for this project which includes the important design of the game such as level design, game logic design, knowledge implementation design and UI design. Moreover, when we proceed to Chapter 5, we will discuss the implementation of the game for this project starting from the installation of the tools, until build-up of the executable game application. This includes an implementation of the game design elements which are selected in the study, along with a detailed exploration of the game design, game elements, and the storyline implemented through the dialogue system within the project particularly using Unity. In addition, we will perform system testing and evaluation for our game application in the Chapter 6. Finally, Chapter 7 provides a summary of the research that provides an overview of this research and some future recommendations.

Chapter 2

Literature Review

2.1 Previous Works on Gamification

Gamification concept is not a stranger word for people especially people nowadays who are born in the 21st century. However, people always misunderstand the game-based learning and gamification concepts and use them interchangeably. Gamification is defined as a series of processes that include many upfront analyses of the study of game elements that apply to certain fields such as finance, medicine, or education while game-based learning is the creation of a game that does not intend to deliver the knowledge to users but merely for entertainment purposes which described in the paper by Kapp [3]. In this paper, we will study what are the important and appropriate game elements that should be included in the game and the implementation of these elements inside the game so that it will deliver the knowledge effectively.

2.1.1 Gamification in Education

For our information, people nowadays favor applying gamification in real-life situations, especially in the educational field where the gamification concept is applied by teachers in school. Jamaludin [4] performed a study on improving learning process via interactive games and reported that gamification and game-based learning enhanced the learning process. Similar result is also reported by Mee Mee et. al. [5], indicated that gamification has the potential to develop creativity, critical thinking, and problem-solving skills for the learner.

Many researchers often studied or found out the truth that what is that the characteristics of gamification because gamification quite often to be related to video games, but it has a non-game context. As proposed by Krath et.al. [6], it can be realized by various methods, which

author suggested that the course matrix can be converted based on levels (just like the levels in the game), experience (attendance and participation in class), complete tasks (such as presentation and talks), fighting monsters (such as exam and quizzes) and so on, so that it can make a normal course more interesting and motivating.

However, it is worth knowing that the game attributes are not always necessary for learning because Landers [8] stated that it will be harmful to learning if the unwanted game attribute is added for gamification. This is the problem known as attribute overlap between serious games and gamification of learning. Landers [7] also suggested that there are a few game attributes there are fantasy, rules/goals, sensory stimuli, challenge, mystery, and control. As the definition provided by Landers [7] in his research, fantasy can be a make-believe environment, scenarios, or characters that involve such as social situations and real-world situations. Moreover, control here means the players will have the ability to control the game, unlike the instruction-based game, in which the player must follow the instructions from the games. Challenge, which is also a game attribute that can make the learning process fun clearly stated goals (which are also integrated with rules/goals attribute) and incremental difficulties. Landers [7] also stated that the mystery is the spaces between known and unknown which create a surprising experience for the gameplay. In this paper, we will have developed games that stimulated the negative effect on obesity, which consider fulfilling these necessary game attributes for gamification.

2.1.2 Gamification in Fitness and Well-being

The gamification concept can be also used in another field other than the education field, which is gamification in fitness and well-being. Its effectiveness in employment helps to enhance the knowledge of people. According to Johnso et.al. [8], the authors suggested that it can trigger intrinsic motivation using the application of gamification that helps in promoting health and well-being. As for that, individuals are encouraged to adopt health-oriented behaviors thereby fostering well-being through the fulfilment of psychological needs through engaging with gamified applications. Notably, Hall [9] outlined crucial game attributes or elements essential for well-being-oriented games. These include a rewarding system for tasks completed by players, as well as an overarching mission that enables point accumulation and progression through different levels. Because of that, this paper aims to develop a game structured around successive levels which could facilitate users' knowledge acquisition across various stages. In addition, a comprehensive mission will be integrated into the game, challenging users to conquer all levels, thereby promoting a holistic engagement with the game's content.

Besides, Mulcahy et.al. [10] pointed out the significance of enjoyment and engagement in influencing players' behaviors within the gamified framework. The authors contend that an excessive emphasis on transformative aspects, such as enhancing well-being and delivering knowledge, can be counterproductive. Therefore, before implementing a game, it becomes imperative to meticulously undertake a process of selecting appropriate elements that strike a balance between different objectives.

2.1.3 Gamification in Healthcare

For gamification in healthcare, we noticed that there exists a greater number of instances which are able to apply the gamification concept in both conceptual and practical usage as compared to the research that we discussed earlier such as conceptual exploration in the fields of education, fitness, and well-being. A notable illustration of gamification within healthcare is presented by Borghese et.al. [11]. In their paper, the gamification concept is employed to monitor and aid in the rehabilitation of stroke patients through the utilization of the Intelligent Gaming Engine for Rehabilitation (IGER). This innovative approach assists patients in consistently engaging in rehabilitation exercises and making use of cost-effective devices. Based on the real-world example of gamification provided by BRADLEY University [12], a video game named SPARX is a game that is specifically designed and made to help treatment for depression which utilized the gamification concepts. Based on the research conducted by Merry et.al. [13], they have proven how effective SPARX is as a tool for treating depression according to their level design which consisted total number of seven levels, each aimed at tackling different aspects of depression. For example, the first level focuses on fostering hope, the second level helps to encourage activity, Level 3 on managing emotions, Level 4 on overcoming challenges, and Level 5 on recognizing negative thoughts. We'll use a similar level-based design approach in this paper to achieve various goals, following the example set by these research studies.

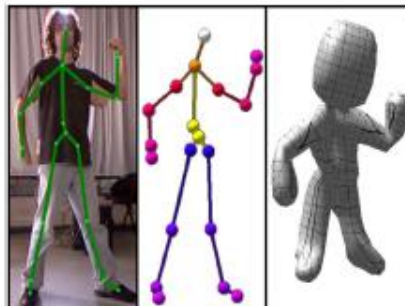


Figure 2.1: The RGB stream, 3D points and final mesh by Johnso et.al. [8]

2.2 Game Design

After examining a range of research papers concerning gamification across different fields, several specific research papers are worth including in this study, which have direct relevance to the game's level design. A study performed by Jiang et.al. [14] reported that obesity and overweight conditions can exert detrimental effects on various organs within the human body, including but not limited to the heart, lungs, blood vessels, liver, and brain. From this spectrum of potential adverse impacts stemming from obesity, a selection of pertinent organ-related effects will be incorporated into the game's level design, forming a central aspect of this paper's exploration. To gain a more comprehensive insight into each organ's suitability at a game level, an in-depth analysis of the effects is undertaken. Another study by JL et.al. [15] suggested that the impact of obesity on the liver is particularly pronounced, leading to the development of fatty liver conditions — a consequence attributed to alterations in carbohydrate metabolism.

In the realm of vascular health, Montani et.al [16] provided the introduction of diet-induced fats not only triggers abnormalities within organs but also results in the accumulation of deposits within major blood vessels. Shifting to the cardiovascular domain, Poirier et.al. [17] highlighted that the escalation of adipose tissue due to obesity, which subsequently elevates the oxygen demand by these surplus fat reserves. As the result, it causes additional strain on the heart, resulting in conditions like hypertension and increased peripheral vascular resistance (SVR). Turning attention to the impact on the brain, Cazettes et.al. [18] reported that obesity induces a state of low-grade inflammation, particularly affecting the hypothalamus and consequently influencing eating behaviours. Building upon these findings, De Souza et.al. [19] and JM and CK [20] further concluded that inflammation within the hypothalamus which results by the obesity, can lead to insulin resistance. Lastly, the influence of obesity extends to pulmonary health as outlined by Costa et.al. [21]. The excess weight associated with obesity

burdens respiratory muscle cells responsible for regulating breathing and managing airway resistance, which can have adverse implications on lung function and overall respiratory efficiency and consequences to asphyxia.

2.3 Summary on Previous Studies

Table 2.1: Comparison Between Gamification in Different Fields

Gamification Field	Education	Fitness and Well-being	Healthcare
Suggested Game Elements	Fantasy, Rules/goals, Sensory stimuli, Challenge, Mystery, Control.	Rewarding system, Point accumulation and Progression through different levels	Level Design for each objective
Advantages of application	Develop learner's creative, critical, and problem-solving skills	Promoting a holistic engagement with the game's content	Treat patient with different stages
Practical implementation of the gamification concept to real system	No	No	Yes
System proposed	Gamification for ordinary school courses	Mobile application for fitness and well-being	SPARX

Table 2.2: Comparison Between Effect of Obesity toward Human Organs

Organs	Negative Effect	Reason
Liver	Fatty liver	Alteration in carbohydrate metabolism
Vessels	Accumulation of deposits	Diet-induced fats
Heart	Hypertension	Escalation of adipose tissue
Brain	Insulin resistance and changes in eating behaviors	Low grade inflammation particularly the hypothalamus
Lungs	Asphyxia	Burdens respiratory muscle cells

2.4 Limitation of Previous Studies

Based on the papers that have been reviewed so far, a significant number of the research papers have primarily centered around conceptual discussions of gamification especially for the papers reviewed in education, fitness, and well-being. However, they often lack in-depth exploration of the practical implementation of gamification concepts and either realise their conceptual ideas into software or video game development except for healthcare gamification. It's important to distinguish gamification in healthcare from its use in educational settings since gamification in learning focuses on enhancing the learning process while gamification in healthcare primarily focuses on assisting patients in recovering from various conditions, rather than just focusing on educational goals which gamification of learning does.

Many researchers have studied game elements and assessed gamification's effectiveness in educational contexts, but this approach fully depends on theoretical studies and statistical correlations. The uncertainty of the practical feasibility of gamification in learning will remain there until game systems or applications are developed which it able to effectively utilize theoretical game elements for educational purposes. We need to prioritize transforming conceptually designed systems into functional games that integrate these theoretical proposed game elements since this approach will not only narrow the divide between theory and practice but also offer a better grasp of how these concepts can effectively drive learning objectives in gamified education.

2.5 Summary

To address the challenges, we outlined in the preceding section we have put forth a comprehensive project that starts from the identification of game elements and attributes to the practical implementation of the gamification concepts into a real functional game system, which particularly focuses on the learning field. The primary objective of this project is to introduce a functional system that utilizes the gamification principles. Our proposed system will employ a structured approach at the game level to progressively deliver knowledge to players about the negative effects of obesity on various human organs. Each level will integrate the established gamification theories and elements, such as challenges, rules, goals, and an element of mystery which we discussed earlier. These selected elements have been proven by well-documented and reputable studies, as substantiated in the previous section 2.1.4.

This level-by-level design is believed to effectively provide useful insights to individuals dealing with obesity-related issues. Also, this approach provides a gamified learning experience through gamification since it is unlike traditional educational methods that are only utilized classically. The goal of this project is to raise awareness among the obese population and the public, by involving them more interactively and engagingly which is supported by findings from Health News of the University of Leeds [23], that demonstrate the considerable impact of in-game advertising on children's health. This proves the effectiveness of this solution and highlights the potential effectiveness of gamified interventions in positively shaping health-related behaviors and disseminating knowledge.

Chapter 3

System Methodology/Approach

This project was organized into different phases including the preliminary study, analysis, design, implementation and testing phases. These phases encompassed the identification and selection of game elements, game design, preliminary study of the tools, resource gathering and the actual implementation of the game, and finally testing and evaluation of the game system.

3.1 System Requirements

3.1.1 Hardware

The hardware used in this project is exclusively a computer device which is the laptop. This computer is dedicated to tasks related to the implementation of the game system, including coding and rigorous testing. The deployment of the system is specific to the Windows operating system, and the computer is employed to validate the deployment process.

Table 3.1.1: Specifications of Laptop

Description	Specifications
Model	HP spectre x360 series
Processor	Intel Core i7-1165G7
Operating System	Windows 11
Graphic	Intel Iris Xe Graphics
Memory	16GB DDR4 RAM
Storage	1TB SSD KIOXIA

3.1.2 Software

In this project, the software that we used is the game engine, which refers to the Unity Engine that we chose. The programming language is C# which can cooperate with the script writing for the game logic of the game engine. The reasons for us to choose the Unity Engine include its accessibility, free of use and availability of online tutorials such as YouTube or Google. Unity also provides the feature for creating 2D games such as the physics system, and control inputs which will be further utilized in this project. The programming IDE that used in this project is Microsoft Visual Studio, which supports the C# editor.

Table 3.1.2: Software and Programming Environment

Game Engine	Unity Game Engine
Version	Unity Hub
Programming Environment	C#
Programming IDE	Visual Studio

3.2 System Methodology

Our project will examine the integration of gamification principles into a well-functioning system based on the principles of Agile Development. This will be divided into different phases such as software specification, software development, software validation and software evolution. In the software specifications phase, the software we will perform the preliminary study of the system, includes the study and identification of the appropriate game elements and tools, learning of the tools and resource gathering, and design of the game such as levels, game logic and UI which are necessary for our project. After we have drafted our ideas, design and requirements, we will use them as the programming guide in the software development phase.

The system will be tested and validated in the software validation phase to examine whether it meets the specific requirements. After we fulfilled all the requirements and validated them accordingly, then the system will be deployed to the OS for the users to access it in the software evolution phase.

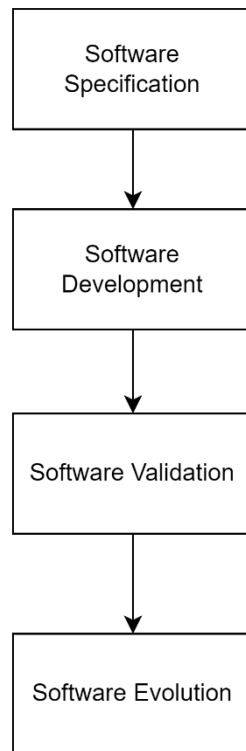


Figure 3.2.1: System Methodology Overview

3.2.1 Software Specification

a) Preliminary study and design of the project

At the beginning of the project, we will thoroughly identify and select the game attributes and elements to be integrated into our game design. This selection process needs to be done under a comprehensive review by preliminary studies of different papers, to ensure that these elements could effectively facilitate the player learning experience. We will then later incorporate these chosen game elements into our game design which aligns with the objectives of this paper. In the current stage of the project, we have outlined the proposed user requirements which are as follows:

- The system should facilitate user learning about the effects of obesity on their body through gamification methods.
- The system should have a controls system that is simple which requiring minimal effort for user to learn.
- The system will implement gamification principles in various aspects including level design, goals, rules, challenges, controls, point accumulation and progression across different levels.
- The system should operate as a standalone application which enabling users to play the game on a local computer without the need for an internet connection.
- The system should have a total number of five distinct levels whereby each of them represent the impact of obesity on different human organs.

In addition to the system requirements, several backlogs have been proposed to meet the system requirements outlined earlier:

Backlog:

- As a player, I want to initiate a new game from the main menu.
- As a player, I want to start the game conversation when starting a new game.
- As a player, I want to start a level after the conversation ends.
- As a player, I want to save my progress upon completing each level.
- As a player, I want to pause the game during gameplay.
- As a player, I want to advance to the next level after completing each one.
- As a player, I want to track the points I have accumulated so far after completing each level.
- As a player, I want to know the gameplay or knowledge in the beginning of each level.
- As a player, I want to know the score obtained after end of each level.

b) System Flow Diagram

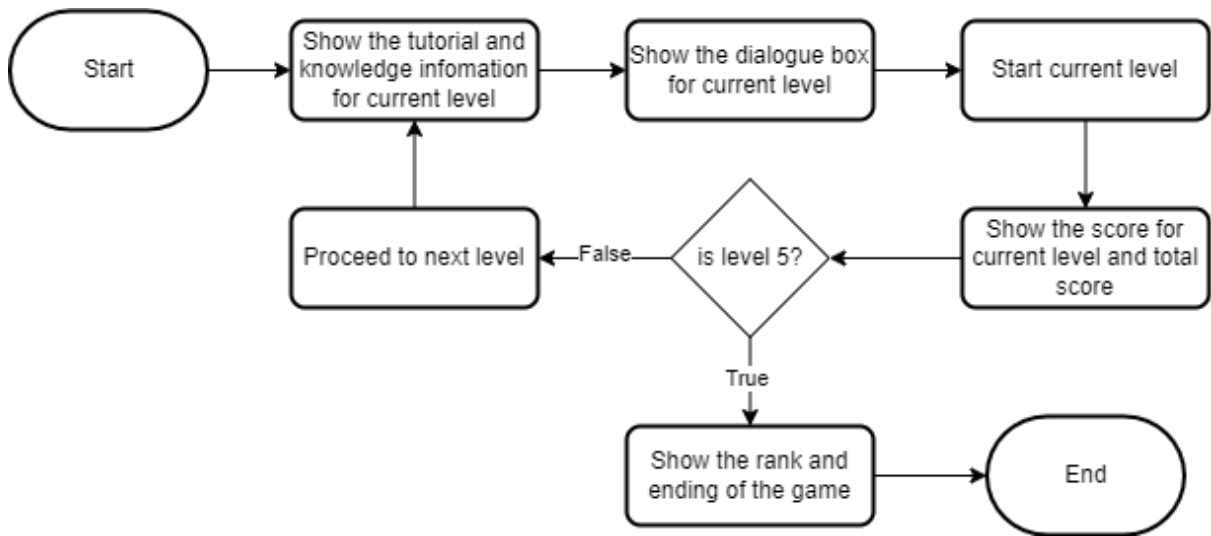


Figure 3.2.2: Flow Chart of the system

c) Assets gathering and creation

To find out the resources and assets needed such as the tile maps for each of the levels, we used the website known as itch.io, which provides the free assets that could be used for unity game development.

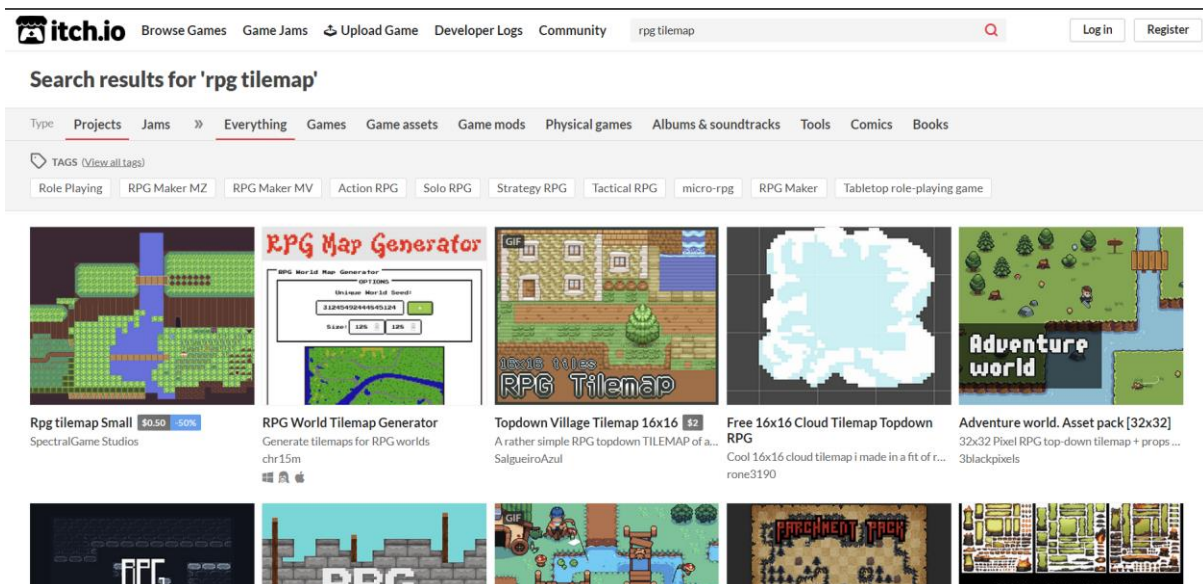


Figure 3.2.3: Assets found from itch.io

Another website for free assets like the button and UI interaction, can be found from the unity Asset Store.

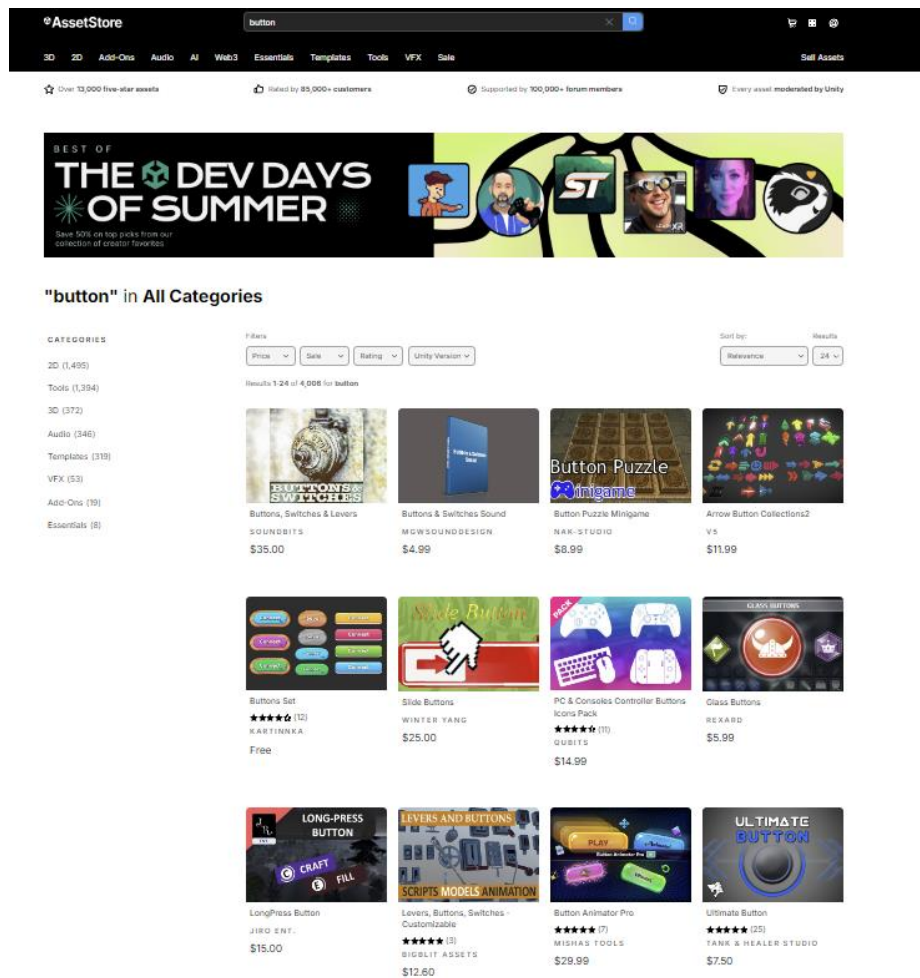


Figure 3.2.4: Assets found from Unity Asset Store

Another way to obtain our assets like the characters, is via the drawing using the Windows Paint, which is also free and easy to use.

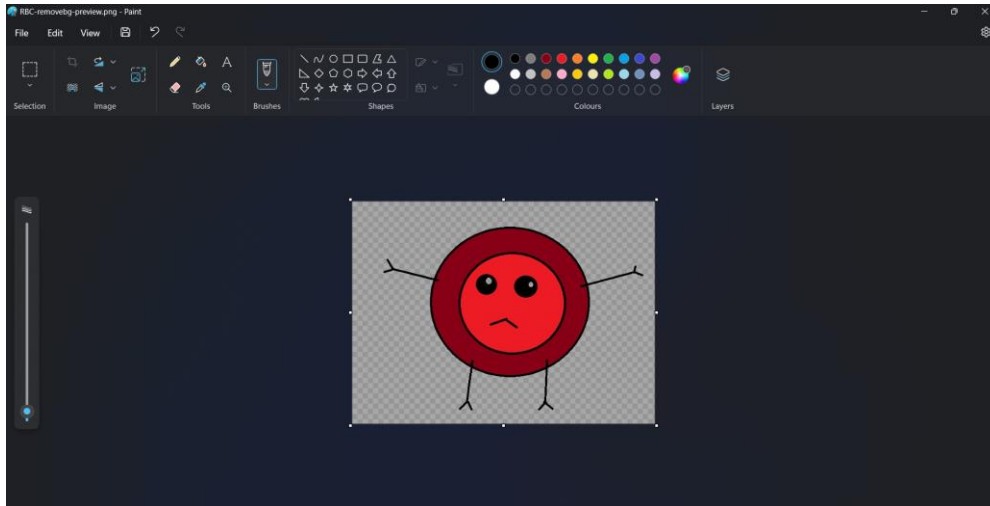


Figure 3.2.5: Assets creation with Paint

3.2.2 Software Development

After we done all the planning, design and resource gathering, we can now move to the next phase which is the software development phase. In this stage, we will start to use the tools which are the game engine, Unity Engine as proposed earlier, and cooperate with the C# editor, Microsoft Visual Studio, as the primary implementation tool for script writing. This application would utilize the OOP concepts, which is the basis for C# programming, and it would easier for us to manage and create based on created class templates.

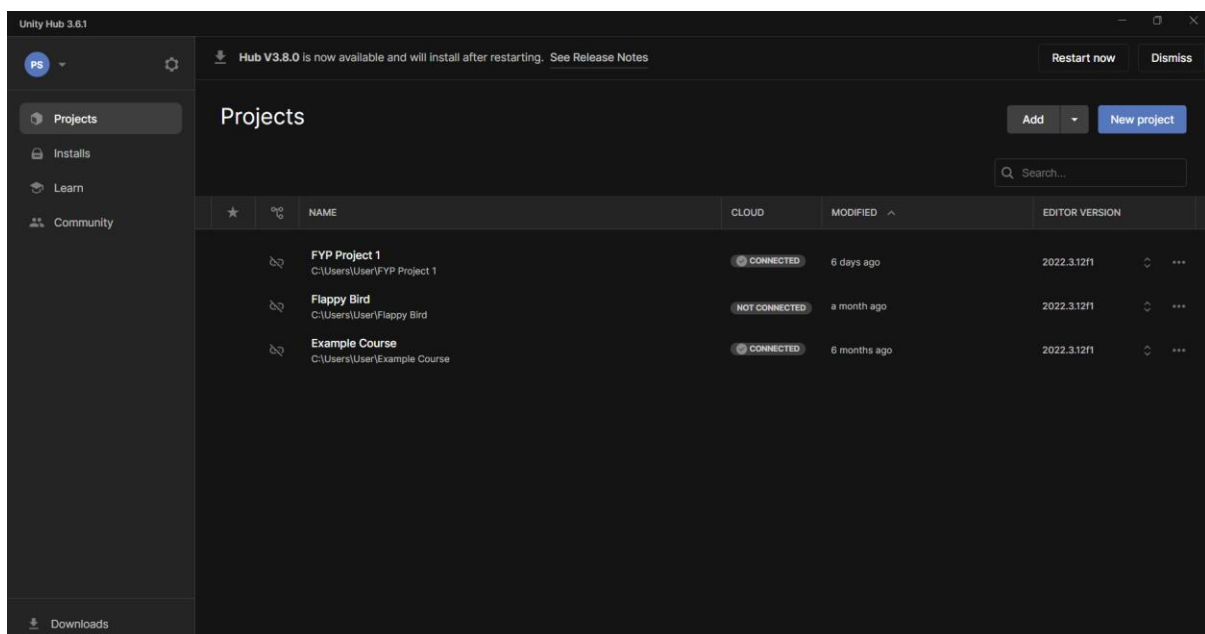


Figure 3.2.6: Unity Hub for unity projects management

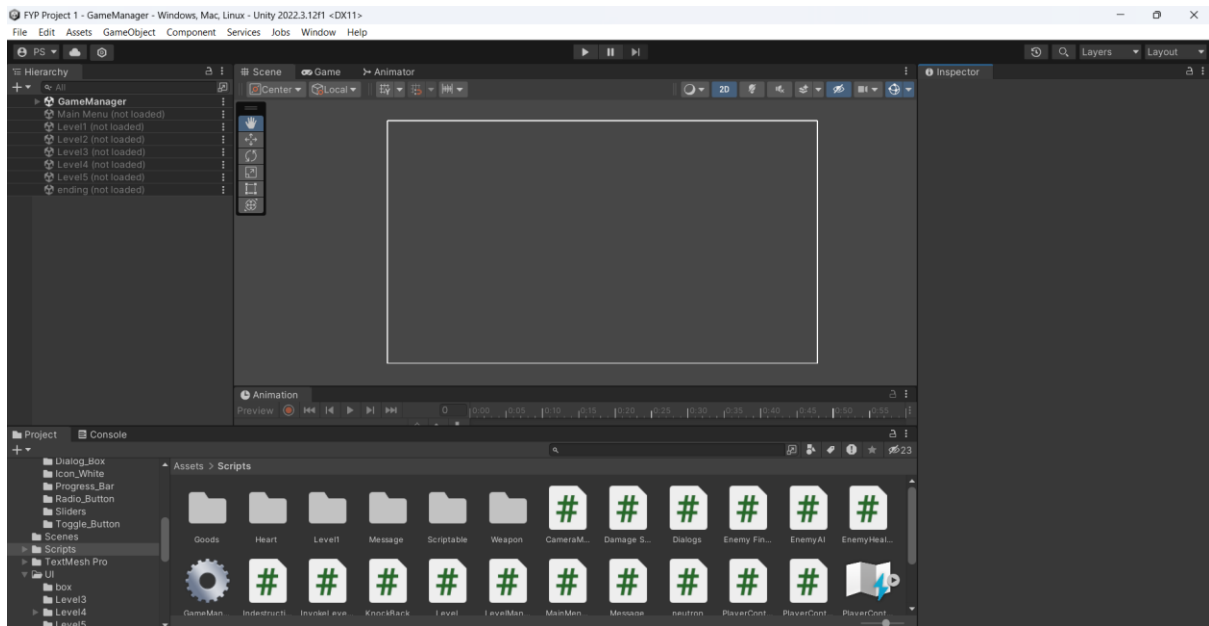


Figure 3.2.7: Unity Editor for game development

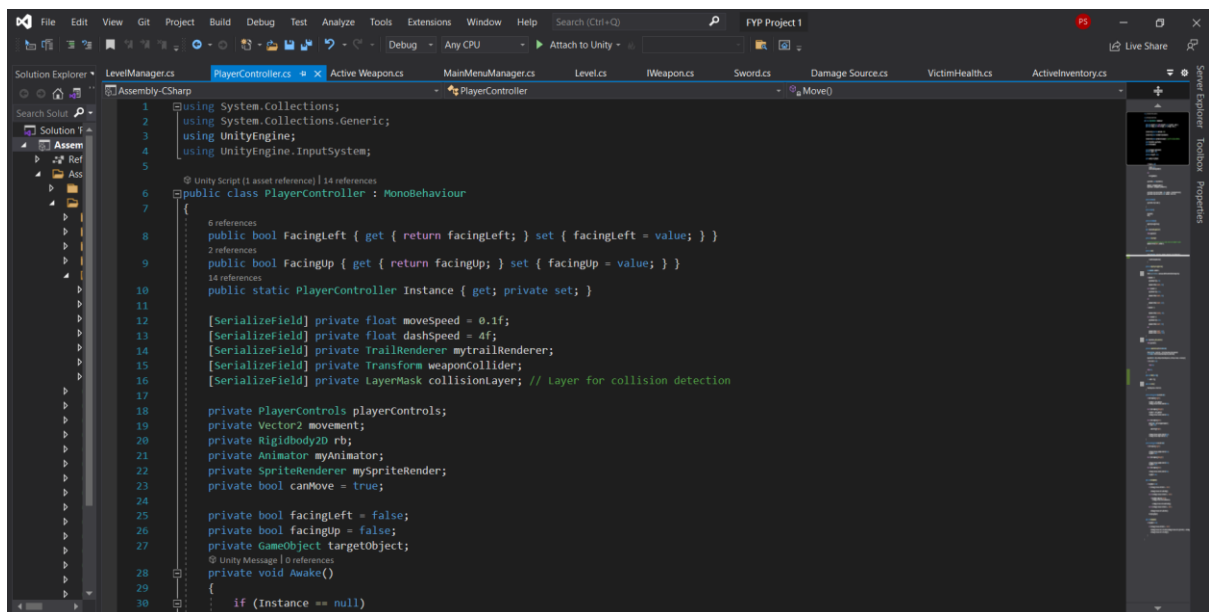


Figure 3.2.8: Visual Studio for C# programming

3.2.3 Software Validation

In this stage, software testing is carried out thoroughly. The testing of the system should cover all the requirements which stated in the user requirements and the backlogs. The system will also be enhanced during this phase to ensure that the developed system is functioning well. The requirement in this project includes all the gameplay elements such as the player movement, flow of the game from one level to the next level, score calculations and knowledges projected to the player.

3.2.4 Software Evolution

At this final stage, the software will be deployed on the operating system, to make it accessible for the end-users. Moreover, the system will continue enhanced and improved to incorporate new requirements and the alterations that will occur in the future works.

3.3 Timeline

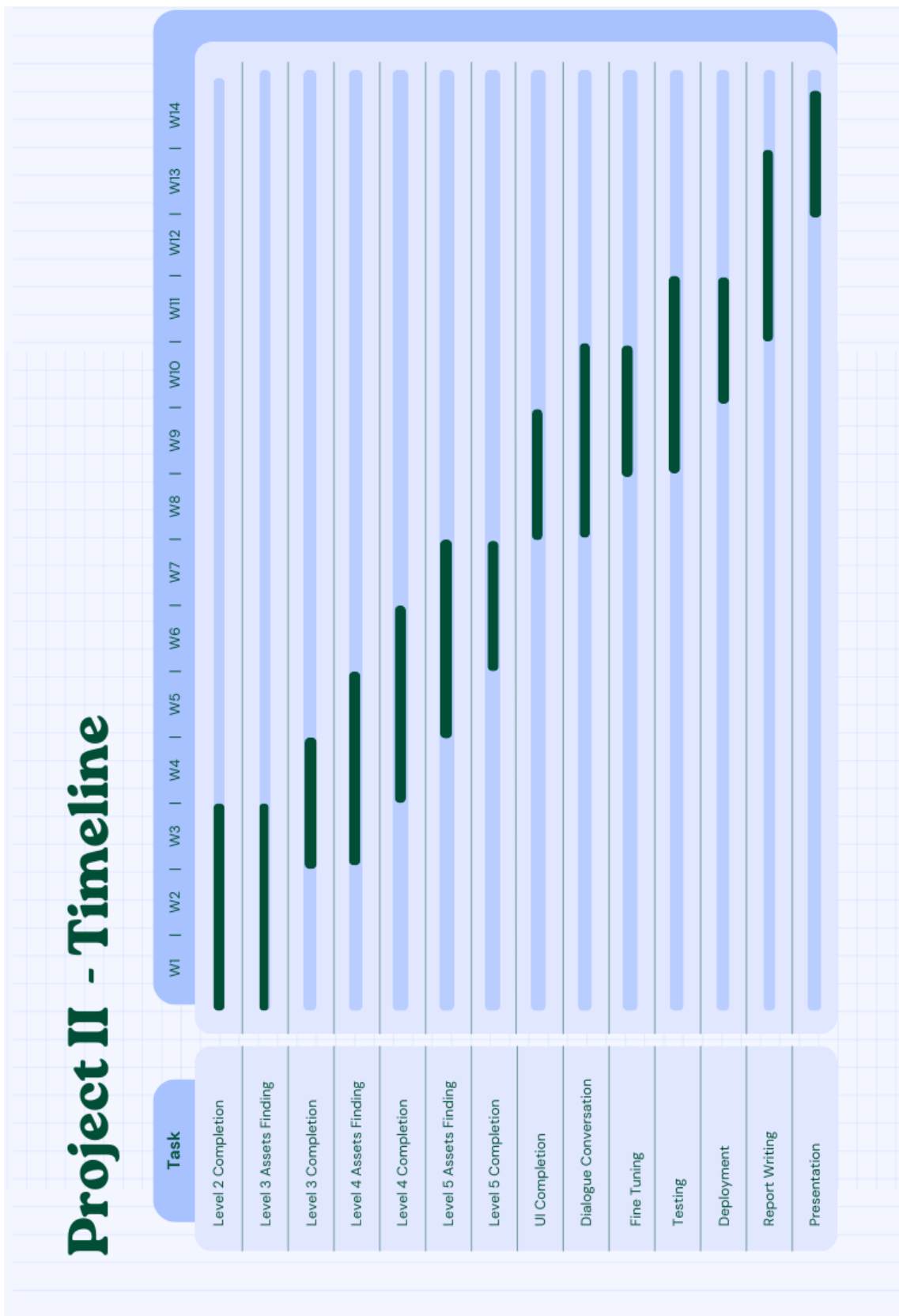


Figure 3.3: Timeline for Project II

Chapter 4

System Design

In this chapter, we will clearly explain the system design for our game application which includes the system block diagram, the system components specifications, the system logic and the integration of knowledge with the game.

4.1 System Design Diagram

4.1.1 System Block Diagram

The system block diagram below describes the overall system design and working flow in general.

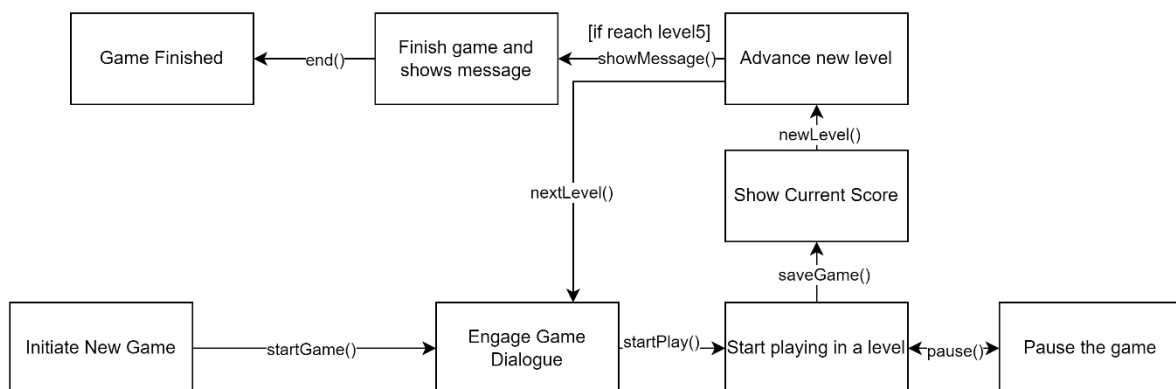


Figure 4.1.1: System Block Diagram

4.1.2 Activity Diagram

The activity diagram below shows the process of the system starting from the beginning of the game until the end of the game.

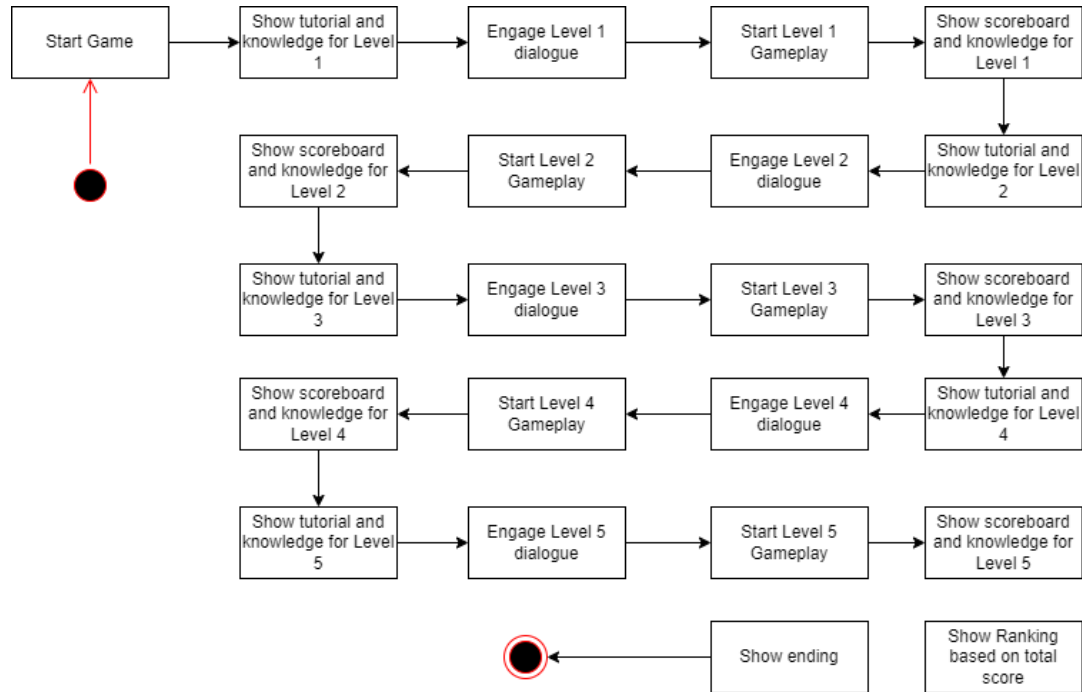


Figure 4.1.2: Activity Diagram of the system

4.1.3 System Hierarchy Diagram

The system hierarchy diagram below describes the game object hierarchy for each of the scenes.

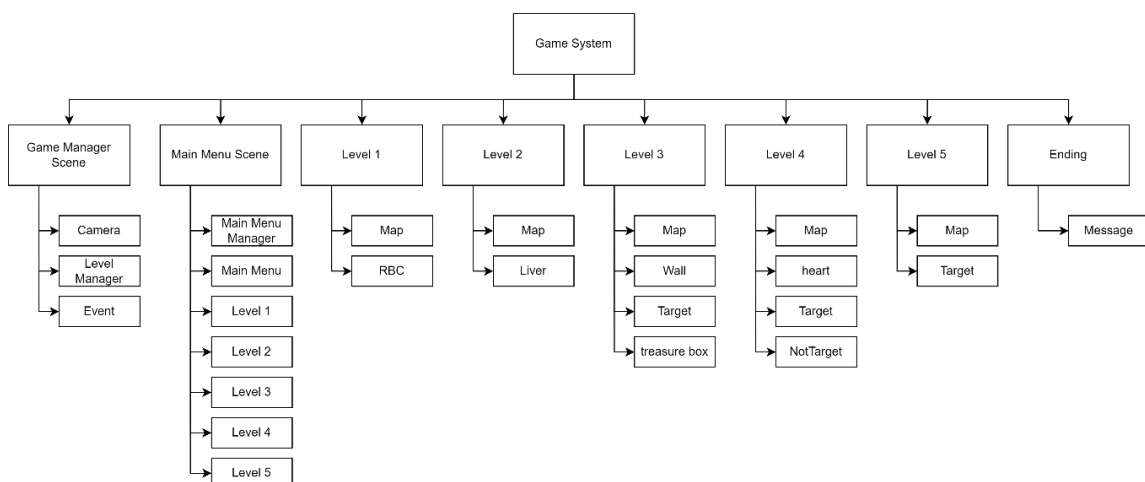


Figure 4.1.3: System Hierarchy Diagram

4.2 System Components Specifications

Our system consists of different levels and scenes which can be further subdivided into different parts, including the components inside different levels, the player and enemies, the maps of different levels, the UI design for the menu, and the game flow of the system.

4.2.1 Levels

The levels design in this project is focused on the different effects of obesity towards human organs, which are blood vessels, liver, lungs, hearts, and brain. Each of these levels will have different gameplay such as fighting with the enemy and saving the victims. Below is the level description for each of the levels:

a) Level 1

Characters:

Bad Characters (Enemy):

- Cholesterol (yellowish guys)

Victim:

- Red Blood Cell (reddish guys)

Brief Scenario:

The player starts with the reddish place, which is the blood vessel as described, the red blood cells start to talk with the player and ask the helps from the player to clear the yellowish guys which represent the cholesterol.

Objective:

The player's objective is to clean up the yellowish substances (cholesterol) that cause the block of the vessel, to move into the red blood cells to become normal. The player must chase and attack the cholesterol which is the yellowish guys using provided weapons such as fruits and foods.

Weapons:

- Sword

Scoring:

- +1 points for defect each yellowish guy
- -1 points for penalty if hits victim

b) Level 2: Liver

Characters:

Bad Characters (Enemy):

- Fats

Victim:

- Injured Liver cells

Brief Scenario:

After player finished the level 1, player will then go to level 2 which is the liver. In this level, player meets the liver cells, and they asked player to help to treat them, at the same time, help them to clear up the fats inside the liver.

Objective:

The player's objective is to save the injured liver cells and restore the liver to health. The player must utilize weapons that treat fatty liver disease, understanding the long-term consequences of excessive fat storage and irreversible damage to liver cells. Moreover, player can use the appropriate weapons to kill the enemy to get the score.

Weapons:

- Sword
- Heart

Score:

- +1 points for successful treat each of the injured liver cell.
- +1 points for successful kill each of the enemy cell.
- -1 points for each attack on the liver cell.
- -1 points for each wrong treat on the enemy cell.

c) Level 3: Lungs

Characters:

Bad Characters (Enemy):

- None

Victim:

- Lungs cells

Brief Scenario:

After finished Level 2, player will then go to the lungs which is the next level. Player notice abnormalities in the transportation of goods (referring to oxygen and carbon dioxide).

Concerned, player volunteers to assist the victim in resolving the issue.

Objective:

The player's objective is to help with the exchange of oxygen and carbon dioxide in the lungs to maintain normal breathing function. The presence of obesity adds stress to the lungs, making normal breathing more challenging.

Targets:

- Oxygen
- Carbon Dioxide

Score:

- +1 point for successful pick up of the oxygen.
- +2 points for successful delivery of the oxygen into the item box.
- -1 point for wrongly pick up of the carbon dioxide.

d) Level 4: Heart

Characters:

Bad Characters (Enemy):

- Adipose Tissue (AT)

Victim:

- Cardiac muscle cells (Heart cells)

Brief Scenario:

The presence of Adipose Tissue (AT) in the heart area has caused disruption to the acceleration and transfer processes, leading to overheating of the machinery. The player must assist the heart cells by removing the accumulated AT, alleviating the burden on the heart. At the same time, player can pick up the appropriate items that helps the heart cells and heal the injured heart cells.

Objective:

The player's objectives are to eliminate the gathering Adipose Tissue (AT) which is the enemy that interferes with the heart's functions, collect the appropriate items that helps the heart cells and heal the injured heart cells. Obesity contributes to the deposition of fats in the heart, further complicating its ability to perform its vital functions.

Weapons:

- Sword
- Heart

Score:

- +1 points for hitting each AT
- +1 points for heal each injured heart cell
- +1 point for pick up the correct item
- -1 points for hitting each injured heart cell
- -1 points for heal each AT
- -1 point for pick up the incorrect item

e) Level 5: Brain

Bad Characters (Enemy):

- None

Victim:

- Neuron cells

Brief Scenario:

In this final level, player comes to the big brain maze that hiding with neuron cells, each neuron cell will be spread around the maze and player need to find them out based on the given location.

Objective: The player's objective is to find the neuron cells in the brain maze in the given time limit.

Weapons:

- None

Score:

- +1 points for each time successfully found the neuron cell

4.2.2 Player

The player character, referred to as the character that is controlled by the player, is designed to interact with other game objects such as the wall, enemies and the items which trigger the desired functions such as pick up, attack, heal and prevent the player move out of the map. This player game object should consist of the animator that provides the animation for the character, the player controller script that controls the movement, and the active weapons that detect the collision and the input of the player for movement control.

a) Animator

The animator can control the player's game object so the character will face the direction where it moving to. (e.g. When the player moves in the direction of positive x, the character would face the right direction.)

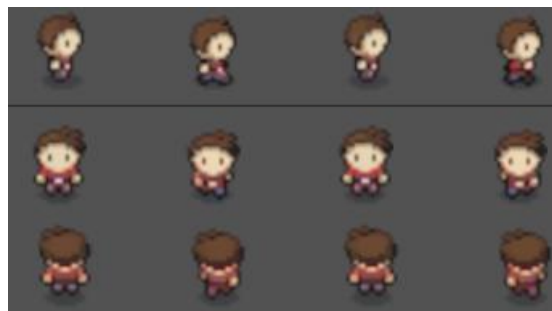


Figure 4.2.1: Player Game Object Animation

b) Player Input

Player input refers to the keyboard input for the player to move the player character over the screen. We would use the common input for character movement, which are W or Up Arrow for up, S or Down Arrow for Down, A or Left Arrow for left, and D or Right Arrow for right. This input component is provided by the unity resources, which means we do not need to define it explicitly and need only the configuration of the control.

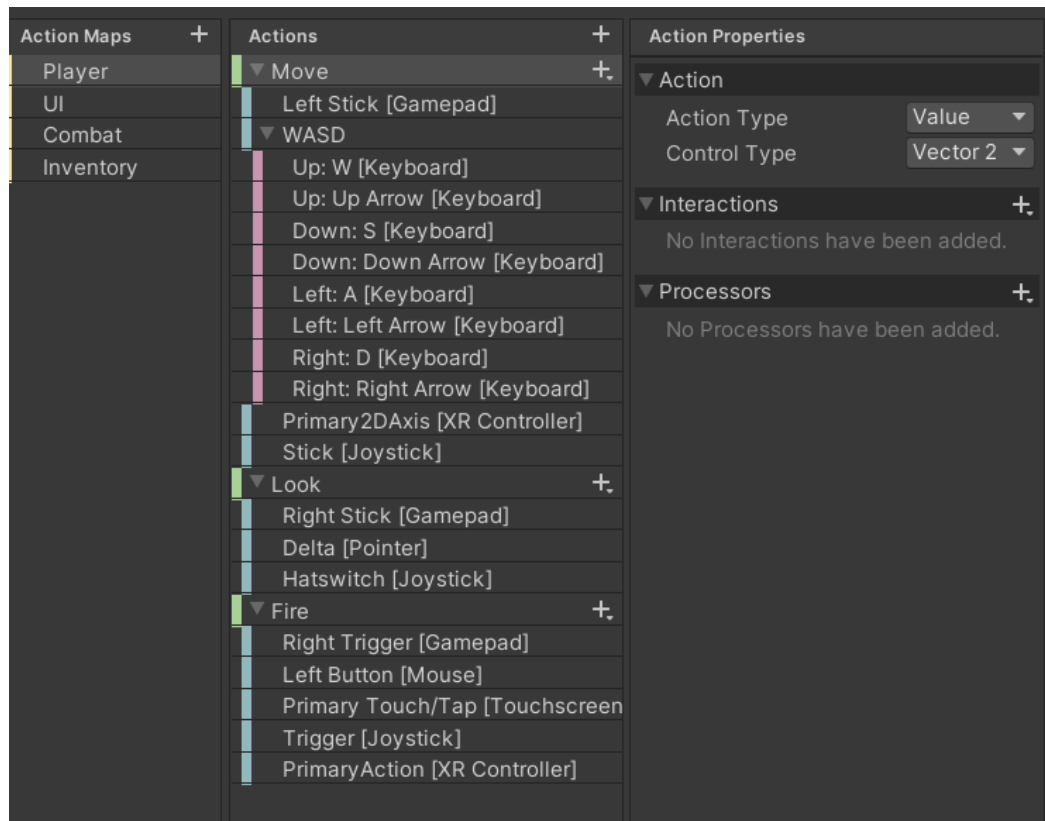


Figure 4.2.2: Player Input Configuration Settings

c) Collider 2D and Rigid Body 2D

Collider 2D referred to the bounding box which defined the size and boundary of the game object, so the script can detect a collision or overlapping when it collides with other game objects. Rigidbody2D refers to the 2D physics as provided by the unity game engine, so when the character moves, or collides, it will follow the physics system such as mass, and gravity which is configured in the settings.

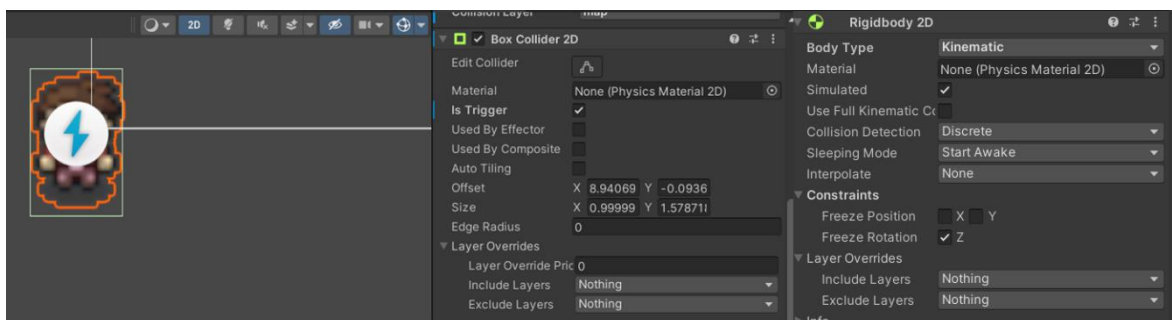


Figure 4.2.3: Box Collider 2D and Rigid Body 2D for Player Game Object

d) Active Weapon

Two types of weapons would be used in this game, which are the sword for attacking and the heart for healing. The logic of changing the weapons and their behavior would be determined by the script as attached to the active weapon game object.



Figure 4.2.4: The Weapons and The Player Game Object

e) Control Script

There are 3 scripts attached to the player game object, which are the player controller script for controlling the movement and its behaviors such as animation and move speed, the damage source which control the logic after causing the damage or collision to other game objects, and the active weapon script that control the logic of the weapons.

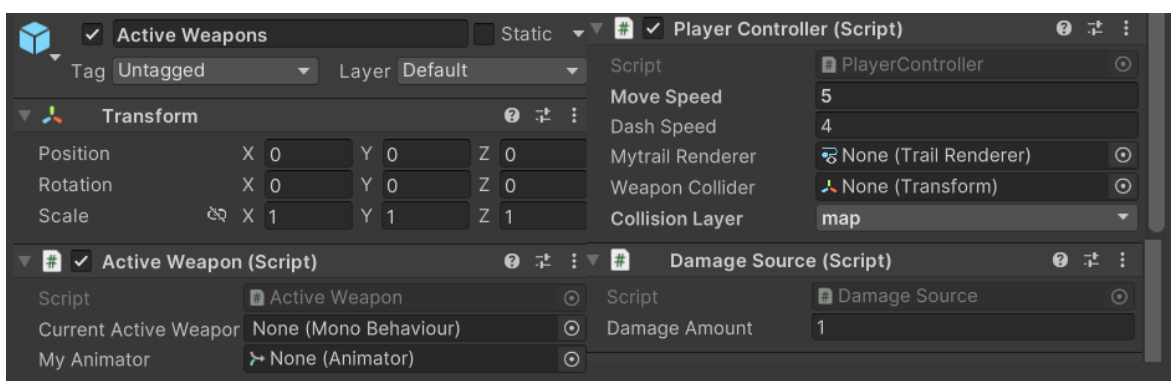


Figure 4.2.5: The Scripts attached to the Player Game Object

4.2.3 Enemies and Victims

The enemies and the victim objects are the game objects that would interact with the player game object which triggered the function like attacking, healing and scoring. The enemies game object would consist of the basic components which are the animator, collider, and scripts for logic control.

a) Animator

The animator of the enemies and victims is used to control the animation where the trigger will happen if a certain condition is met, such as when the health reaches a certain level, and the movement of the enemy.

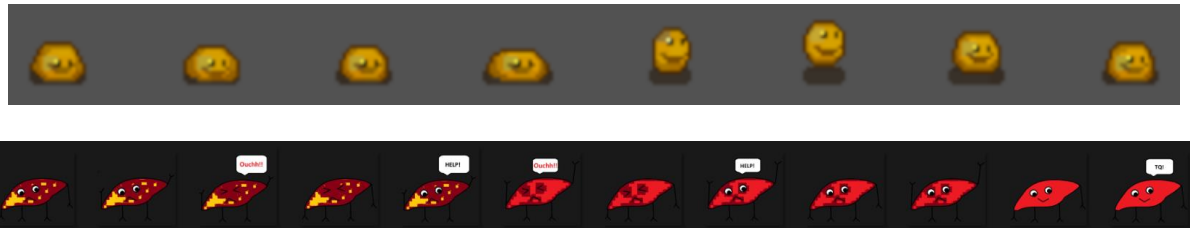


Figure 4.2.6: The Animation for Enemies and Victims Objects

b) Collider 2D

There are different types of colliders used by the enemies and victims' objects, which are polygon colliders and capsule colliders, but their function is the same to detect the collision between objects with the scripts.

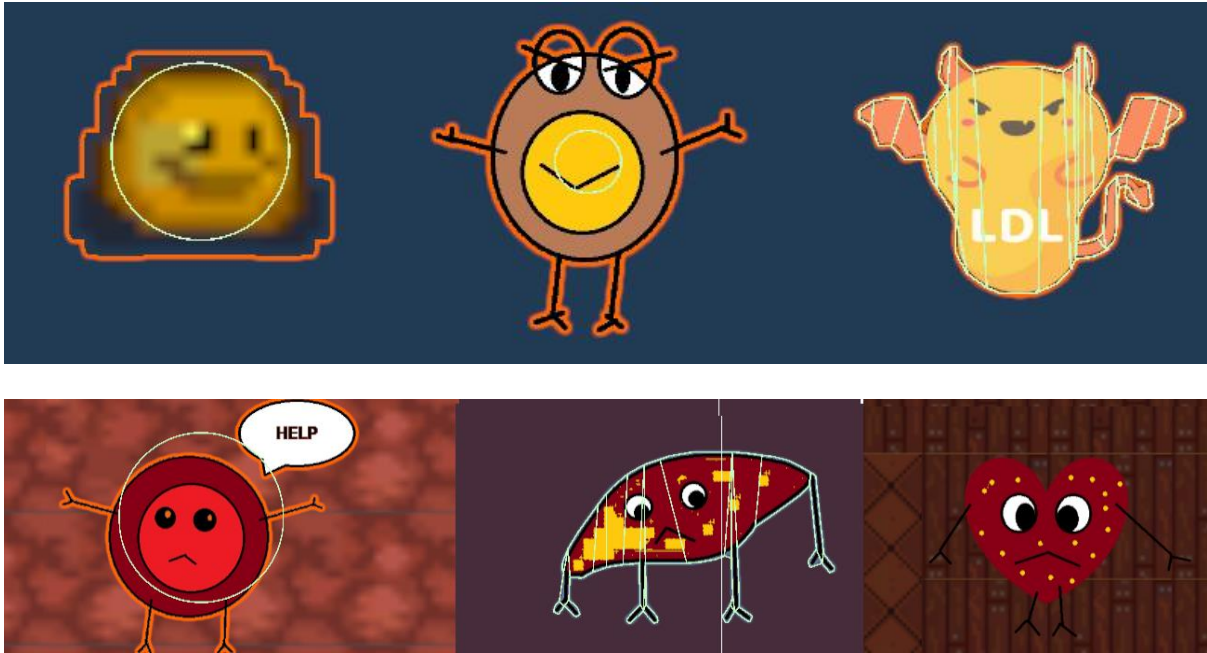


Figure 4.2.7: The Game Objects that Used the Collider 2D

c) Control Scripts

All the enemy objects used the same scripts for movement control and health detection which are the enemy AI, enemy find path, enemy health and knockback scripts. For victim objects, they would use the victim script for health detection and collision detection.



Figure 4.2.8: The Scripts Used by The Enemy and Victim Game Objects

4.2.4 Maps

Maps are the background used for each level of the game and there is a total of 5 different levels that used different of the maps.

a) Level 1: Blood Vessels

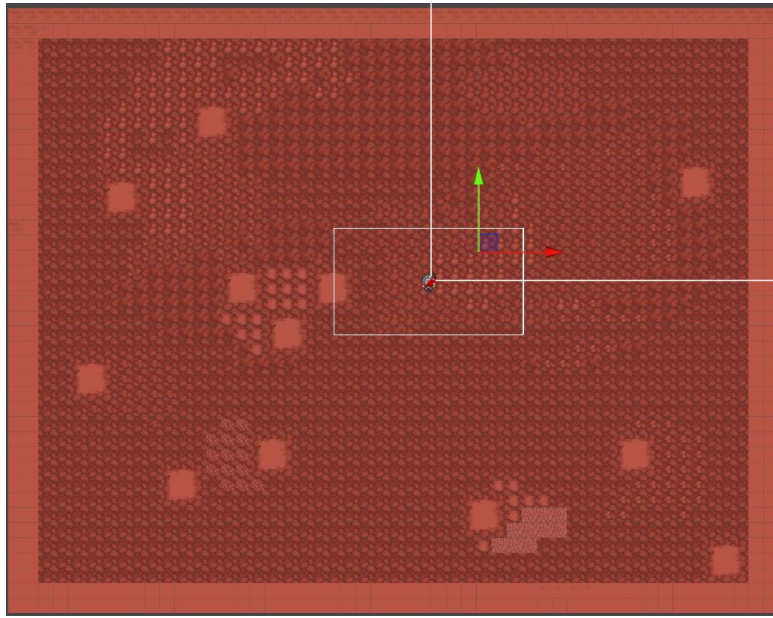


Figure 4.2.9: Map for First Level

b) Level 2: Liver

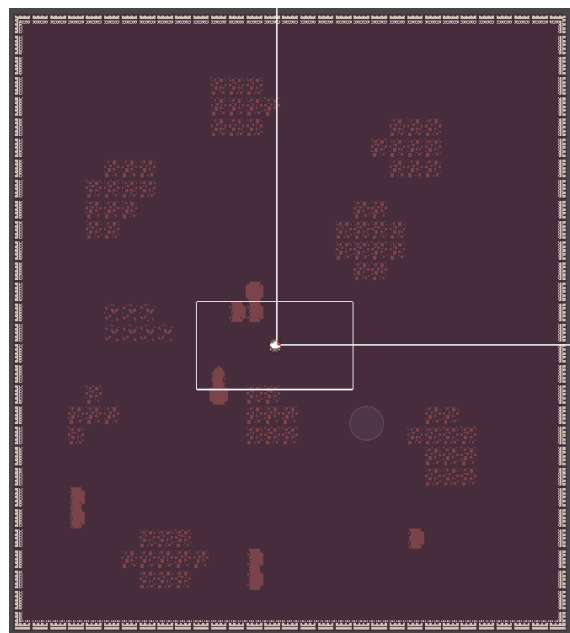


Figure 4.2.10: Map for Second Level

c) Level 3: Lungs

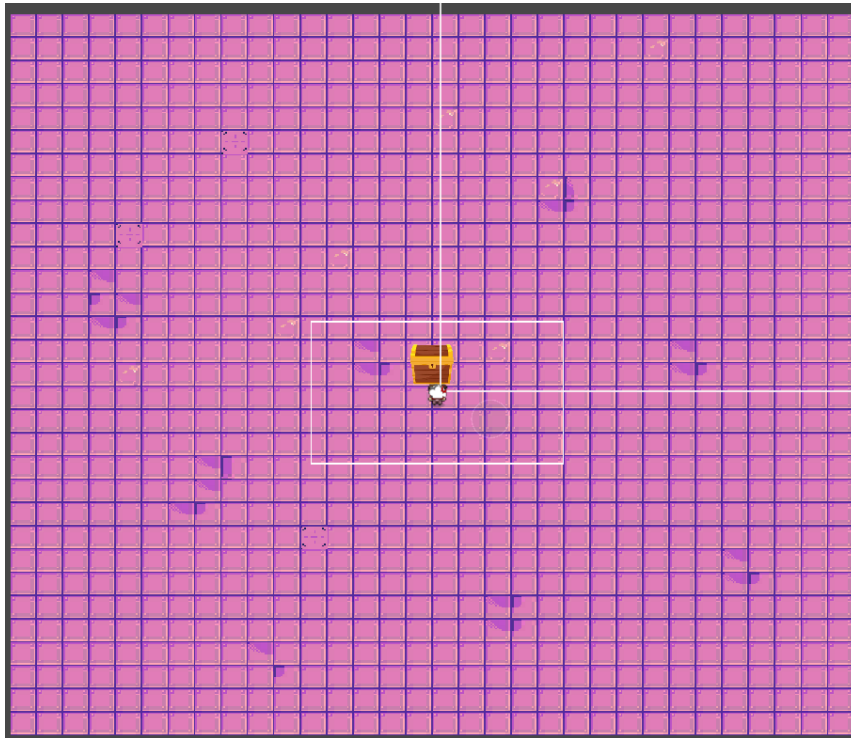


Figure 4.2.11: Map for Third Level

d) Level 4: Heart

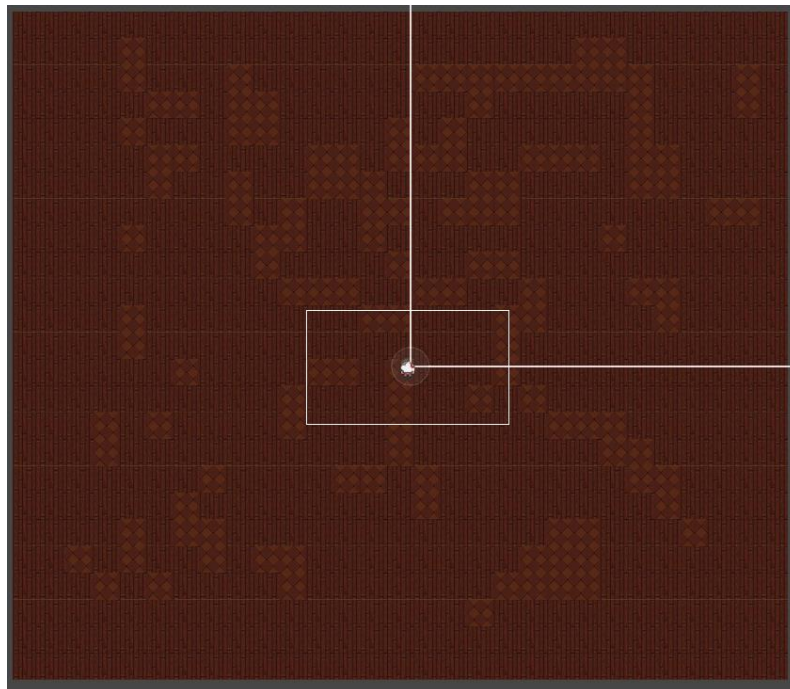


Figure 4.2.12: Map for Level 4

e) Level 5: Brain



Figure 4.2.13: Map for Level 5

4.2.5 Dialogue box and Message Box

The dialogue box is used to represent the conversation between the player and the victim, which is mainly to teach what will player going to do at this level and the reason why the player needs to do so. Moreover, the message box would be the message that we would deliver to the player in the ending part, which represents the “Message for Player”.

Dialogue Box: Level 1

Reddish Guy: Greetings! I'm Reddish Guy, and I'm seeking your assistance in eliminating these pesky yellow creatures that have been causing disruptions to our daily tasks of delivering goods throughout the body.

Reddish Guy: They've overstayed their welcome and are hindering our productivity. Your help in getting rid of them would be greatly appreciated!

Player: Of course, I'm here to help. I'll make sure to clear them out, so they no longer pose a problem for you and your deliveries.

Reddish Guy: Thank you in advance for your assistance!

Dialogue Box: Level 2

Liver Cells: We implore you to aid us! The prolonged effects of obesity are wreaking havoc on our well-being, and without intervention, our fate is dire.

Player: Fear not, I am committed to rescuing each and every one of you from this perilous situation.

Liver Cells: Your assistance gives us hope in our darkest hour. We eagerly await your help in alleviating our suffering and securing our survival.

Dialogue Box: Level 3

Lung Cells: Brave traveler, we are in desperate need of your assistance! The oxygen levels here are dwindling, and without it, we cannot sustain the life-giving process of respiration.

Player: Worry not, for I will gather the oxygen you need. I shall pick it up and deliver it to the box so that your function can continue uninterrupted.

Lung Cells: Your swift action is our lifeline. Please, gather the oxygen and place it in the item box. With your help, we can keep the breath of life flowing through the body.

Player: Consider it done. I will ensure that each bit of oxygen reaches you safely.

Lung Cells: Your dedication fuels our hope. We eagerly await the arrival of the oxygen, trusting in your ability to save us from this crisis.

Dialogue Box: Level 4

Heart Cells: Brave one, we need your help! The yellowish enemies are attacking us, and we're growing weaker by the minute. Our survival depends on you!

Player: Don't worry, I'll take care of those enemies. But tell me, is there anything else I can do to help you recover?

Heart Cells: Yes, please! We're hurt and in desperate need of healing. There are special items nearby that can restore our health. If you can pick them up and bring them to us, we might just survive this ordeal.

Player: I'm on it. I'll remove the enemies and find those healing items. Hang in there, help is on the way!

Dialogue Box: Level 5

Neuron Cells: Adventurer, we're in a dire situation! Some of our fellow neurons have gone missing within this vast brain maze. Without them, our communication network is breaking down!

Player: A missing neuron network? That sounds serious. How can I help?

Neuron Cells: We need you to navigate through this complex maze and find the missing neurons. Their exact locations were revealed in the previous screen. Only you can bring them back to restore our full function.

Player: Understood. I'll retrace the information from the previous screen and track down the missing neurons. I won't let you down!

Neuron Cells: Thank you! Your help is crucial to restoring the brain's full capability. We trust in your skills to bring our lost comrades back.

Message Box: Ending

This game serves as a poignant reminder of the detrimental impact of obesity on our vital organs, such as the blood vessels, liver, lungs, heart, and brain. Through the lens of this gameplay experience, players gain insight into the grave consequences that obesity can inflict upon the body.

Drawing from real-world data, statistics from the National Health Screening Initiative 2023 reveal that a staggering 53.5% of Malaysians are affected by obesity. This alarming prevalence underscores the urgency of raising awareness and promoting healthier lifestyles within our communities.

As we conclude this journey, let us carry forward the lessons learned, and the awareness gained. Let us strive to make informed choices that prioritize our health and well-being. By adopting healthier habits and supporting one another in our efforts, we can work towards combating obesity and safeguarding the health of future generations.

May this game serve as a catalyst for positive change, inspiring individuals to take proactive steps towards a healthier, happier life. Together, let us stay aware, stay healthy, and forge a brighter, more vibrant future for all.

4.2.6 UI design

a) Button

The buttons used in this project are the buttons that are used to skip the current page such as a tutorial, knowledge information and score page which are represented by the blue rectangular button, and other buttons are used in the options menu during the game.

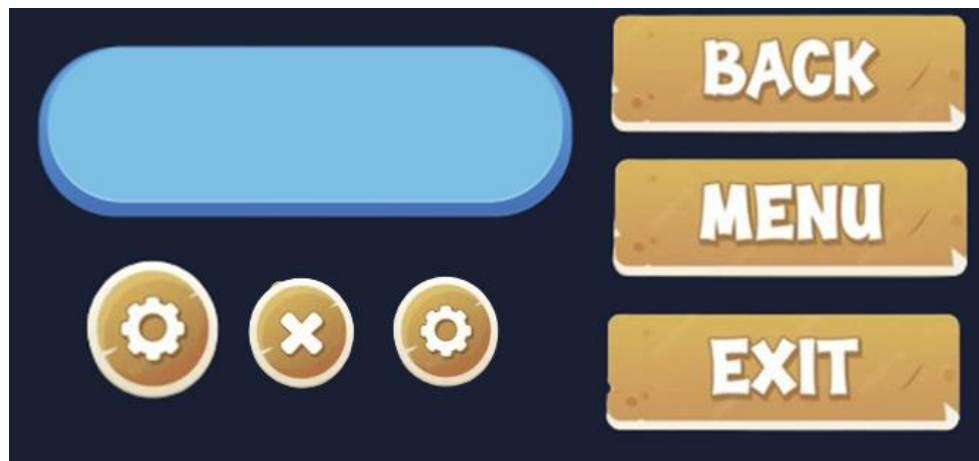


Figure 4.2.14: Buttons used in the game

b) Menu

There are two menus in this game: one is the option menu, and the other is the main menu which appears on the first page when the user enters the game. All the buttons in this menu are based on the design as previously.



Figure 4.2.15: Menus in this game

d) Scoreboard

The scoreboard is used to record the time and the item picked by the player, which is designated as shown below:

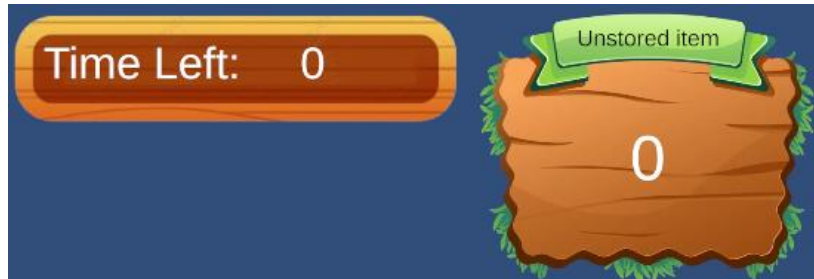


Figure 4.2.16: Scoreboards in this game

e) Information

In this project, we would have the introduction of the level page, knowledge page, score page and the total score page for each level like the example of level 1 below.

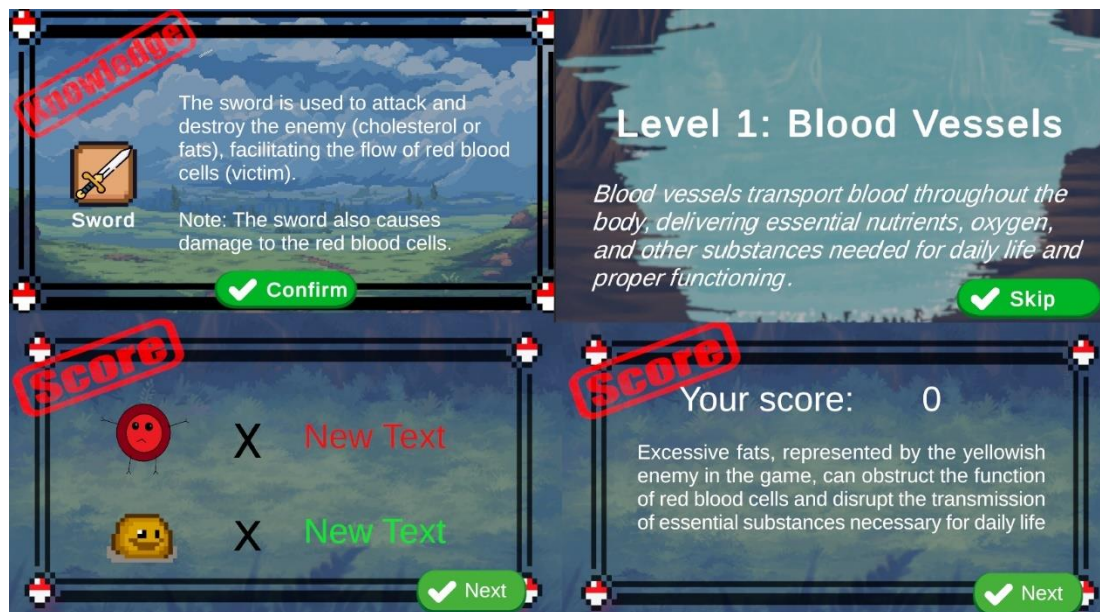


Figure 4.2.17: Example of pages design in Level 1

f) Dialogue and Message Box

The dialogue box would prompt the player once the player enters each level and the player can only start to play once the dialogue box is finished.

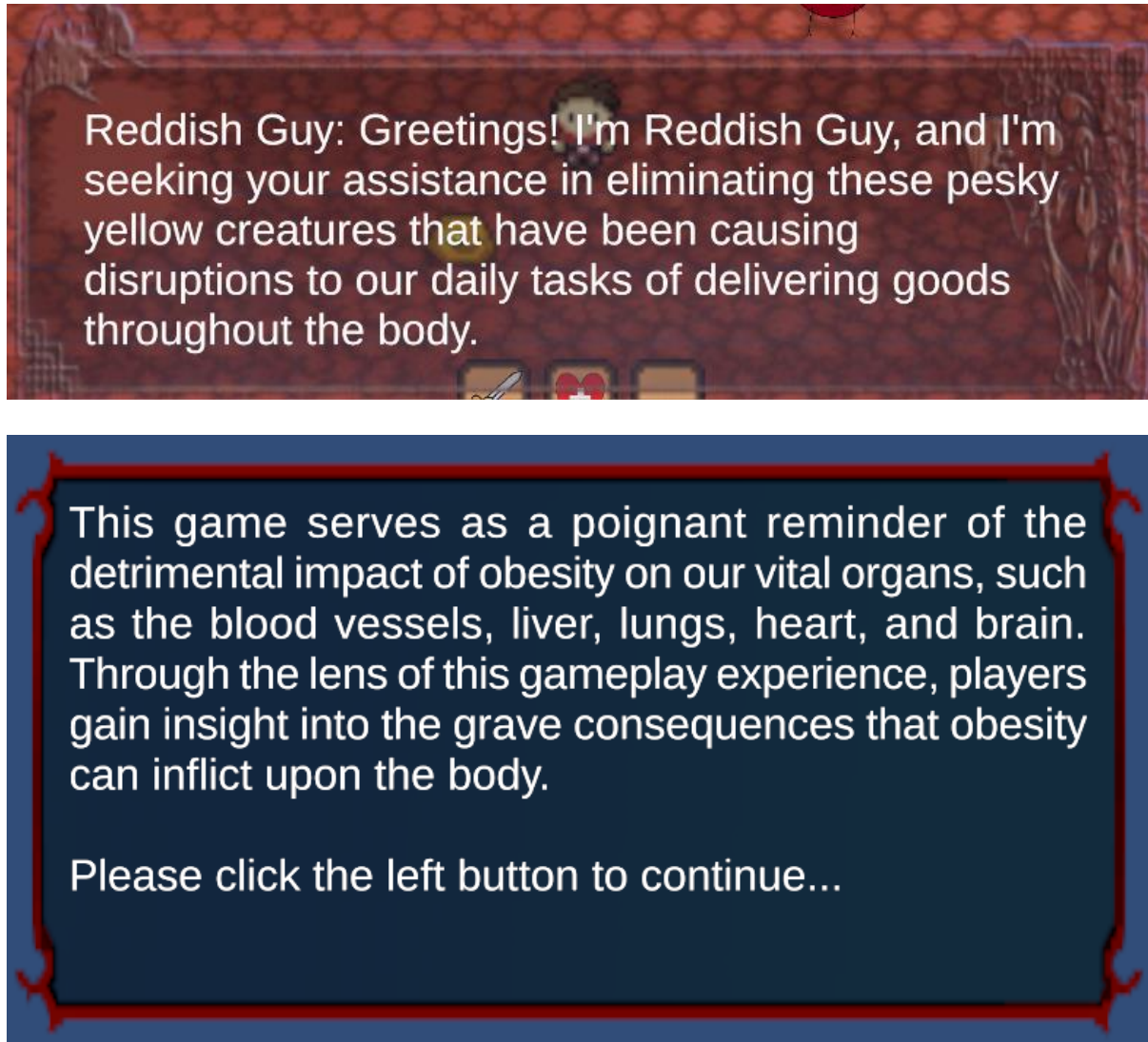


Figure 4.2.18: Dialogue box and Message Box in the game

4.2.7 Ranking

The ranking refers to the rank or the level of understanding that judges the player could understand well the knowledge delivered by the game. The player ranking can be judged based on several levels of score.

Table 4.2.1: Ranking Classification Based on Score Range

Score	Rank	Description
100 and above	Perfect	Outstanding performance! You fully understand the effects of obesity on body organs and their functions, demonstrating comprehensive knowledge throughout the game.
80-99	Great	Impressive! You have a strong grasp of the impact of obesity on the body's organs and functions, missing only a few minor details.
60-79	Well	Good effort! You show a reasonable understanding of the effects of obesity on organ functions, but there are still areas to improve.
40-59	Satisfactory	Decent work. You have a basic understanding of how obesity affects the body, but there are several aspects that require further attention.
39 and below	Unsatisfactory	There's room for improvement. Your understanding of the effects of obesity on body organs and functions is limited, and further study is recommended.

4.3 System Logic

The entire game logic is controlled by the level manager and main menu scripts where the slides or page are only shown to the player when certain conditions are met. Moreover, all the data flow in the game is stored inside the level manager object and defined in the level manager-script. Therefore, all the score and rank of the player would be determined via the gameplay and classification via the defined script. The slides or pages for each level and main menu can be classified and stored respectively as shown below.

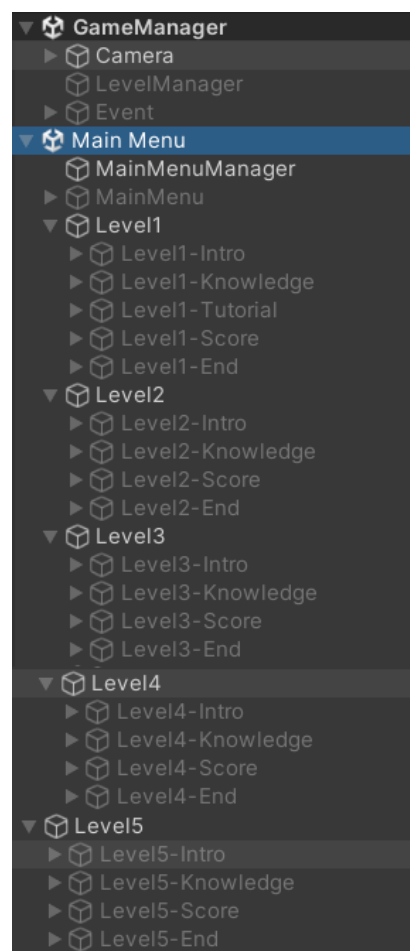


Figure 4.3.1: The structure of the game manager scene and main menu scene

Figure 4.3.1 shows the game objects hierarchy for both game manager and main menu scene. Game manager scene would control the logic of the game, such as the game scoring, level flow while main menu scene would control each of the slides or pages before and after each of the level. These two scenes contain the scripts that are the core logic for this game which are described below:

a) Level Manager Script

Functions:

- Awake(): Initializes the singleton instance and finds UI elements.
- Start(): Starts the game from level 1 and sets up the initial timer.
- Update(): Updates the timer for the current active level, checks if the level is passed, and transitions to the next level if completed.
- IsLevelPassed(): Checks if the level is passed based on the getPass() method.
- ChangeToNextLevel(): Handles transitioning between levels, unloading the current level, and loading the next.
- setIsDialog/getisDialog: Manages the dialogue state for conversations within the game.
- resetTimer(): Resets the timer for the current level.
- openOption/closeOption/exitOption: Manages the game's option menu.
- calScore/getScore: Calculates and returns the overall player score across levels.

Key Features:

- Singleton Pattern: Only one instance exists during the gameplay
- Level Management: Manages five different levels using custom Level class objects.

- Timer: Manage the time for each level, ensure each level within the desired duration.
- Score calculation: Tracks and calculates the score of players based on the performance in each level.
- UI interactions: Manages UI elements such as buttons, inventory, scoreboards and the option menu.
- Dialogue state management: Manages the status of the dialogues to display and undisplay the dialogue box.

b) Main Menu Manager Script

Functions:

- Awake(): Initializes the singleton instance and finds UI elements.
- StartGame(): Starts the game from level 1 and shows the tutorial canvas.
- QuitGame(): Quits the application or stops play mode in the editor.
- Level1Skip(): Skips to the knowledge canvas for level 1.
- Level1SkipKnowledge(): Skips the knowledge canvas and activates the level manager.
- Level1Start(): Begins level 1 by transitioning to the intro canvas.
- Level1Next(): Moves from the score canvas to the end canvas in level 1 and updates the score.
- Level1End(): Ends level 1 and transitions to the intro canvas for level 2.
- Level2Knowledge(): Skips to the knowledge canvas for level 2.
- Level2Start(): Begins level 2 and sets the level to active with an appropriate scene load.

- Level2Next(): Moves from the score canvas to the end canvas in level 2 and updates the score.
- Level3Intro(): Prepares for level 3 by showing the intro canvas.
- Level3Knowledge(): Skips to the knowledge canvas for level 3.
- Level3Start(): Begins level 3 by setting the level to active and loading the scene.
- Level3End(): Transitions to the end canvas in level 3 and updates the score.
- Level4Intro(): Prepares for level 4 by showing the intro canvas.
- Level4Knowledge(): Skips to the knowledge canvas for level 4.
- Level4Start(): Begins level 4 by setting the level to active and loading the scene.
- Level4End(): Transitions to the end canvas in level 4 and updates the score.
- Level5Intro(): Prepares for level 5 by showing the intro canvas.
- Level5Knowledge(): Skips to the knowledge canvas for level 5.
- Level5Start(): Begins level 5 by setting the level to active and loading the scene.
- Level5End(): Transitions to the end canvas in level 5 and updates the score.
- ToEnding(): Transitions from level 5 to the ending scene.
- LevelEnd(): Handles ending the current level and transitioning to the score canvas based on the current level.

Key Features:

- Singleton Pattern: Ensures only one instance exists throughout gameplay.
- Level Management: Manages transitions and progression between levels 1 to 5.

- **Score Calculation:** Updates the score dynamically based on player performance in each level.
- **UI Interactions:** Controls the visibility and transitions of various UI elements (menus, canvases).
- **Scene Loading:** Manages scene loading for different levels using additive mode.

4.4 Integration of knowledges

This section refers to the knowledges that would be delivered to the player in this game for each level. The knowledges are delivered progressively in each level and would be shown to the player before and after during the gameplay.

Level 1: Function of blood vessels and effect of obesity on blood vessels

Function: Blood Vessels

Blood vessels transport blood throughout the body, delivering essential nutrients, oxygen, and other substances needed for daily life and proper functioning.

Effect:

Excessive fats, represented by the yellowish enemy in the game, can obstruct the function of red blood cells and disrupt the transmission of essential substances necessary for daily life

Level 2: Function of liver and effect of the obesity towards liver

Function: Liver

In fat metabolism, liver cells break down fats and produce energy.

Effect:

Excessive fats, represented by fat cells in the game, can obstruct the function of liver cells, which might lead to the death of these cells.

Level 3: Function of lungs and effect of the obesity towards lungs

Function: Lungs

The lungs are responsible for the exchange of gases, including the intake of oxygen and the expulsion of carbon dioxide, playing a crucial role in maintaining respiratory function.

Effect:

Excessive fat accumulation in the lungs can obstruct their function, making it harder for them to perform gas exchange.

Level 4: Function of heart and effect of the obesity towards heart

Function: Heart

The heart is the central organ responsible for pumping oxygen-rich blood from the lungs to the entire body.

Effect:

Obesity can impair heart function, as the accumulation of fats puts additional strain on the heart.

Level 5: Function of brain and effect of the obesity towards brain

Function: Brain

The brain plays a crucial role in regulating and controlling hormones, particularly in the hypothalamus, which are essential for maintaining the processes of fat absorption and synthesis in the body.

Effect:

Obesity can impact the brain, leading to low-grade inflammation, particularly in the hypothalamus, which controls hormone regulation.

Chapter 5

System Implementation

In this chapter, we will discuss the implementation of this project, including the hardware setup, software setup, setting and configuration before the project, the complete system operation, implementation issues and challenges, and concluding remarks.

5.1 Hardware Setup

In this project, the hardware we used to develop and test for the program is just our laptop, and the hardware specification can be shown below:

Table 5.1: Specifications of Laptop

Description	Specifications
Model	HP spectre x360 series
Processor	Intel Core i7-1165G7
Operating System	Windows 11
Graphic	Intel Iris Xe Graphics
Memory	16GB DDR4 RAM
Storage	1TB SSD KIOXIA

To improve the development workflow and increase productivity, we also used an external mouse and keyboard. This setup enhanced precision during testing and allowed for a more comfortable and efficient development experience.

5.2 Software Setup

This section refers to the preparation of the software before starting with the development process, which includes the unity account registration, unity hub and unity editor installation, and finally the visual studio installation.

5.2.1 Unity Account Registration

To register an account with Unity, we need to go to the URL: <https://id.unity.com/> and choose to create one account. After clicking the create one, we need to fill in our details for registration.

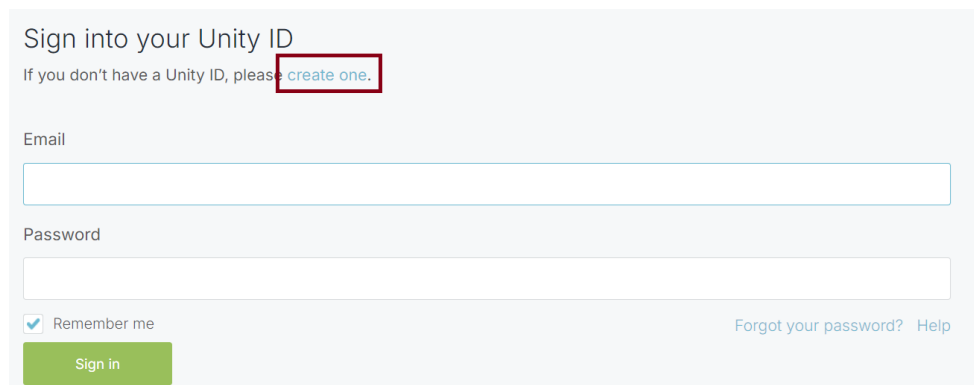


Figure 5.2.1: The login page of the Unity ID

After clicking the *create one*, we need to fill in our details for registration.

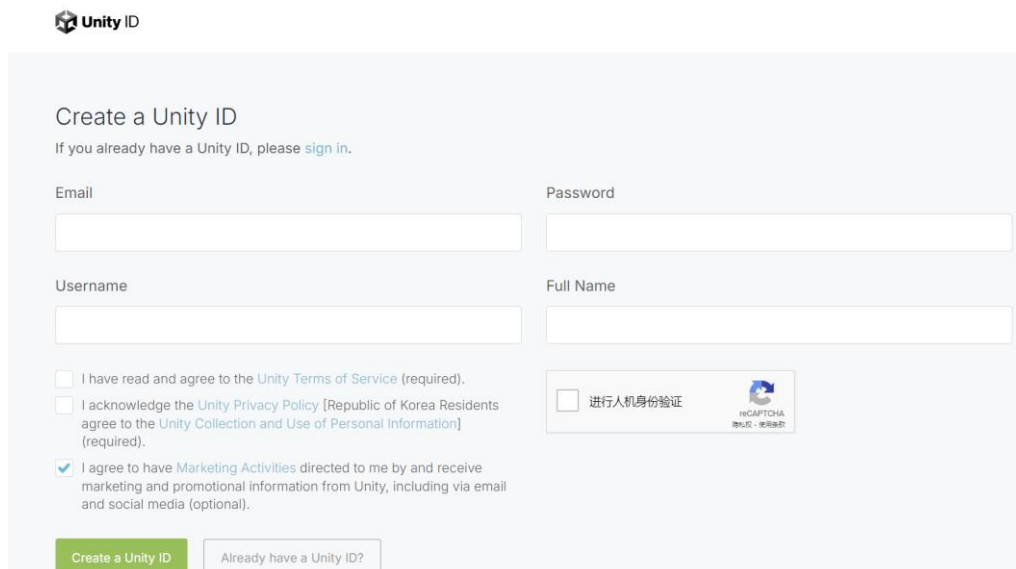


Figure 5.2.2: The registration page of the Unity ID

After we did the registration process, we were ready to download the software in the next section.

5.2.2 Unity Hub and Unity Editor

To download the Unity Hub, we need to go to the website: <https://unity.com/download>, where we will be asked to download an executable file, namely *UnityHubSetup.exe*.

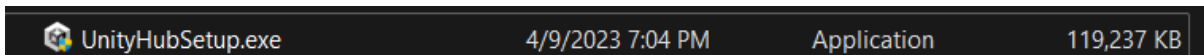


Figure 5.2.3: The downloaded executable file

After the executable file is downloaded, just execute the file and we will be asked to install the Unity Hub, and we can follow the instructions to install the program.

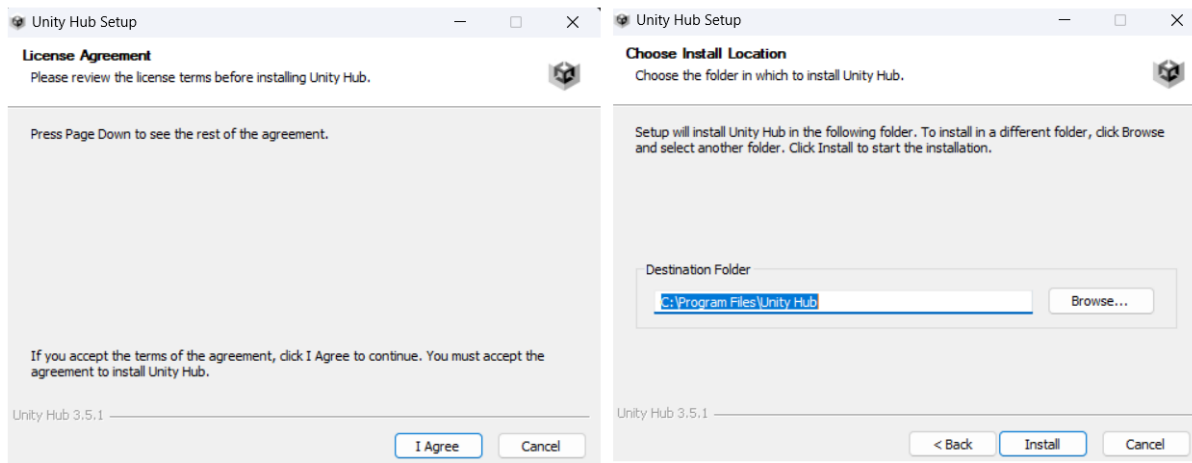


Figure 5.2.4: The Unity Hub Installation Process

After the installation process is completed, then we can log in to our newly registered account as in section 5.2.1, as shown in the figure below:

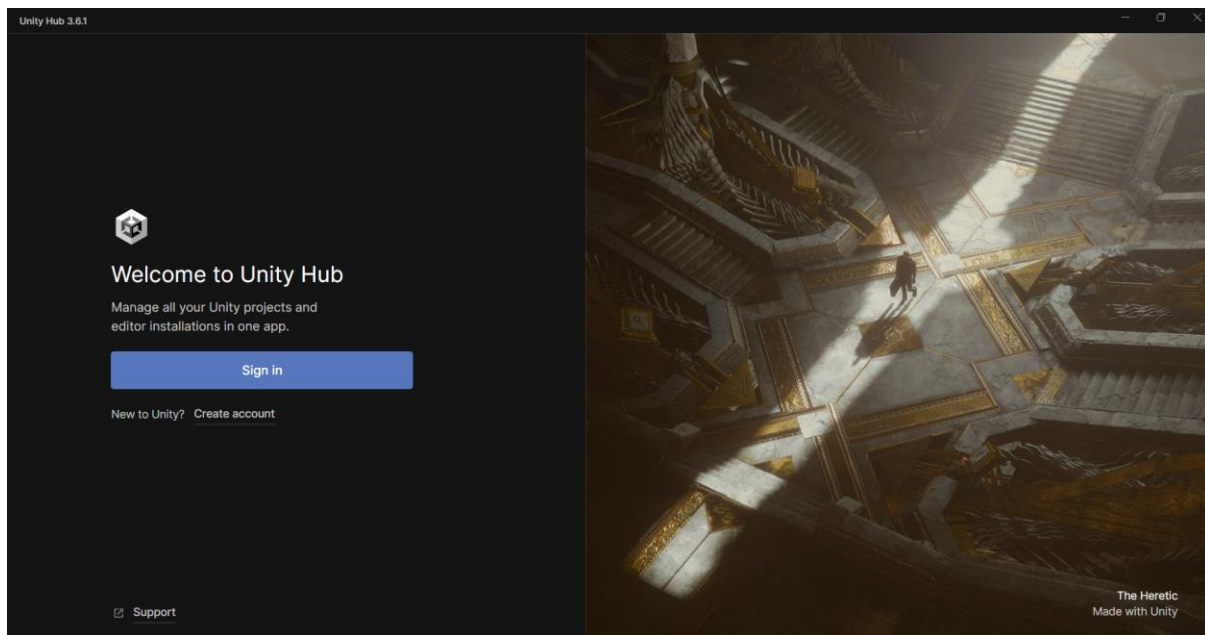


Figure 5.2.5: The Unity Hub Login Page

After we have logged in, we should see the page below:

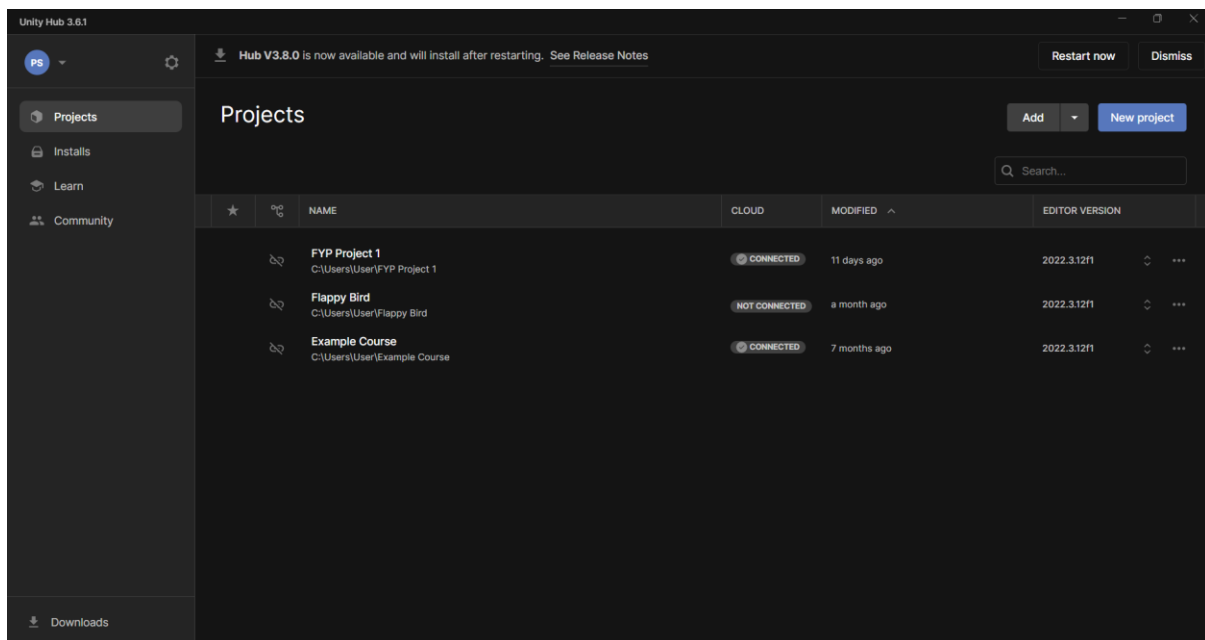


Figure 5.2.6: The Unity Hub Page After Successful Logged In

In this stage, we can proceed to install the Unity Editor by clicking the *Installs* in the left panel, and we should notice that the button to let us install the Unity Editor.

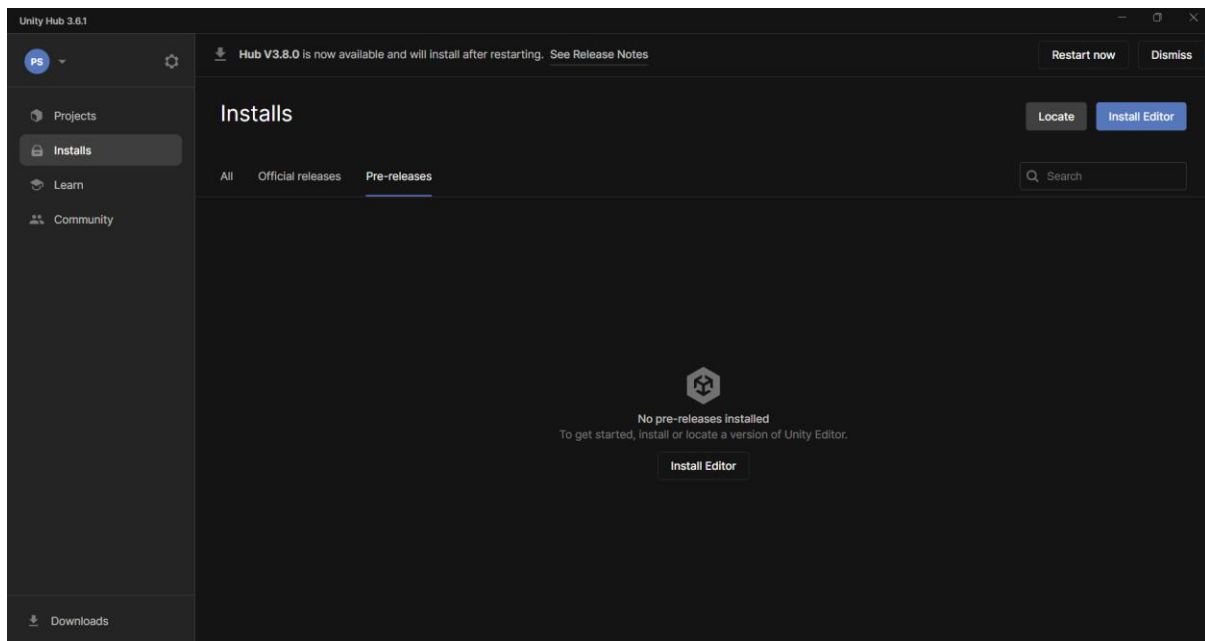


Figure 5.2.7: The Installs Page in Unity Hub

After the Unity Editor had been successfully installed, we did the setup for Unity and proceeded to install Visual Studio in the next section.

5.2.3 Microsoft Visual Studio

To install Microsoft Visual Studio, we can visit the Microsoft official website, <https://visualstudio.microsoft.com/>, to download the free version of Visual Studio.

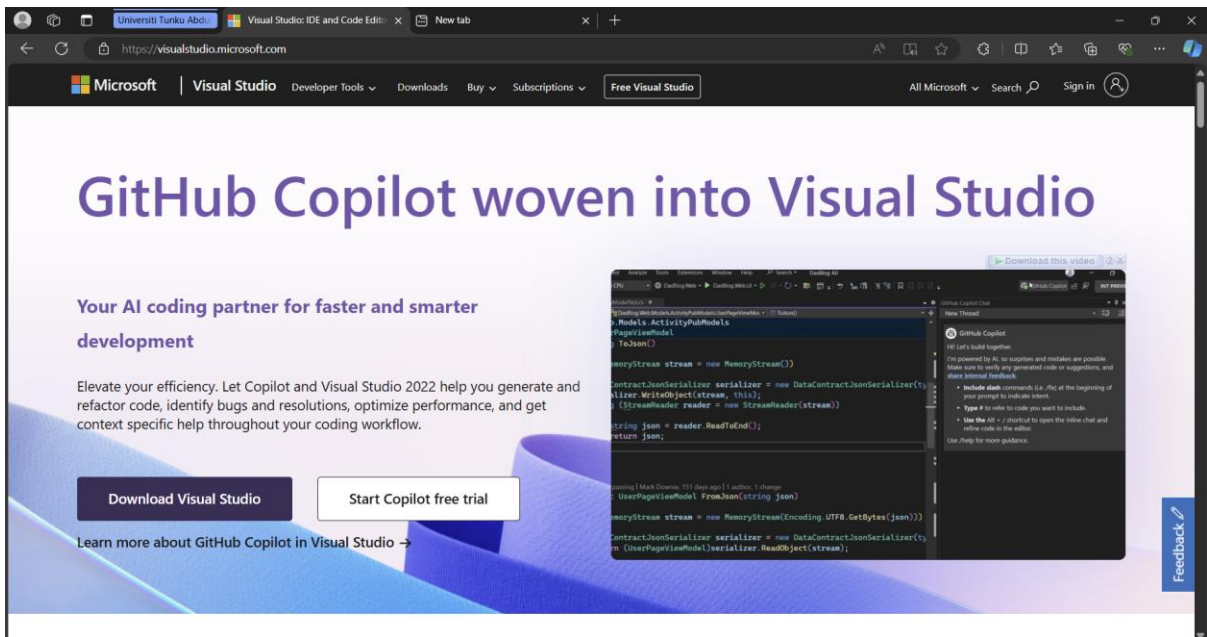


Figure 5.2.8: The Visual Studio Download Page

After we have downloaded the executable file, we can proceed to install the program.

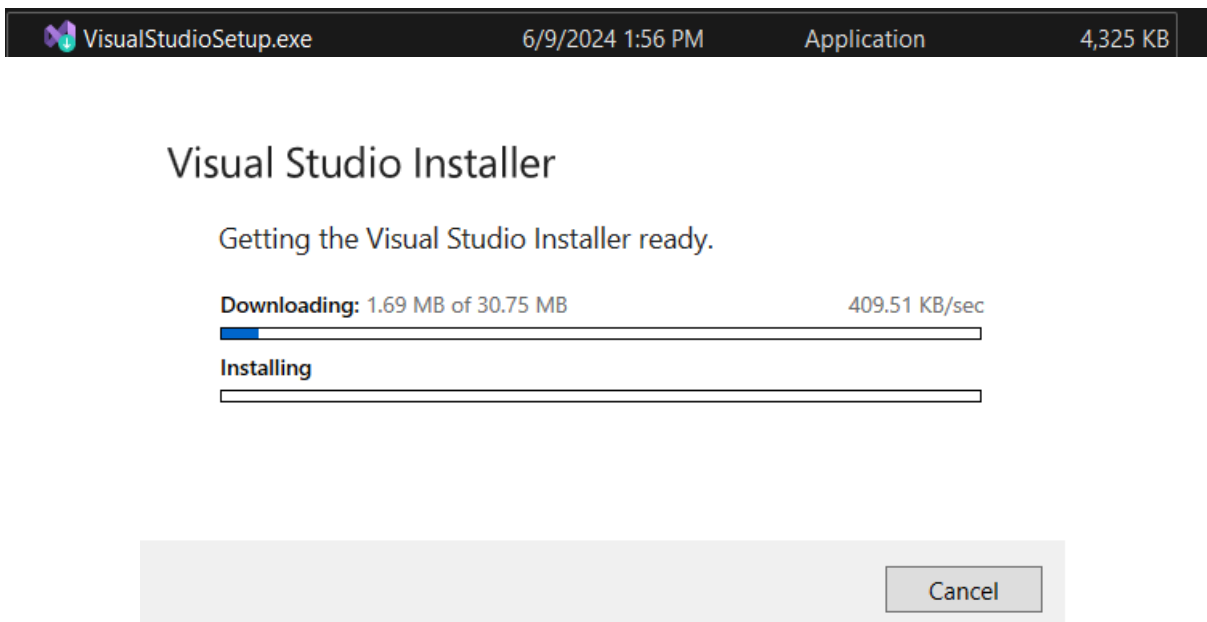


Figure 5.2.9: The Downloaded Executable File and the Visual Studio Installation Process

When we finished the installation process, we should have successfully installed the Visual Studio as our development IDE depends on the version of the Visual Studio. In this project, we would use the Visual Studio Enterprise 2019 as our IDE.

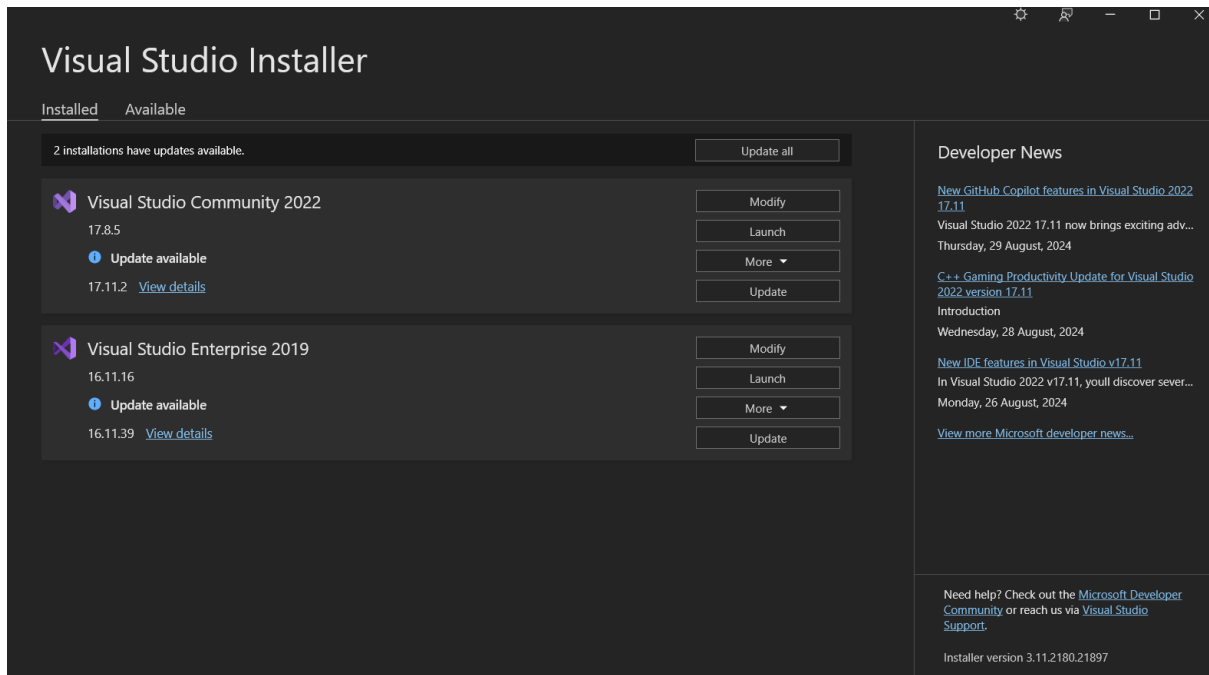


Figure 5.2.10: The Installed Version of the Visual Studio

5.3 Setting and Configuration

To implement this project, it is important to know the way how to create a new project from Unity Hub, the settings of the project, and some of the features provided by Unity.

5.3.1 Create a new project from Unity Hub

To create a new project, launch the Unity Hub program and click the “*New project*” button as shown in the figure below.

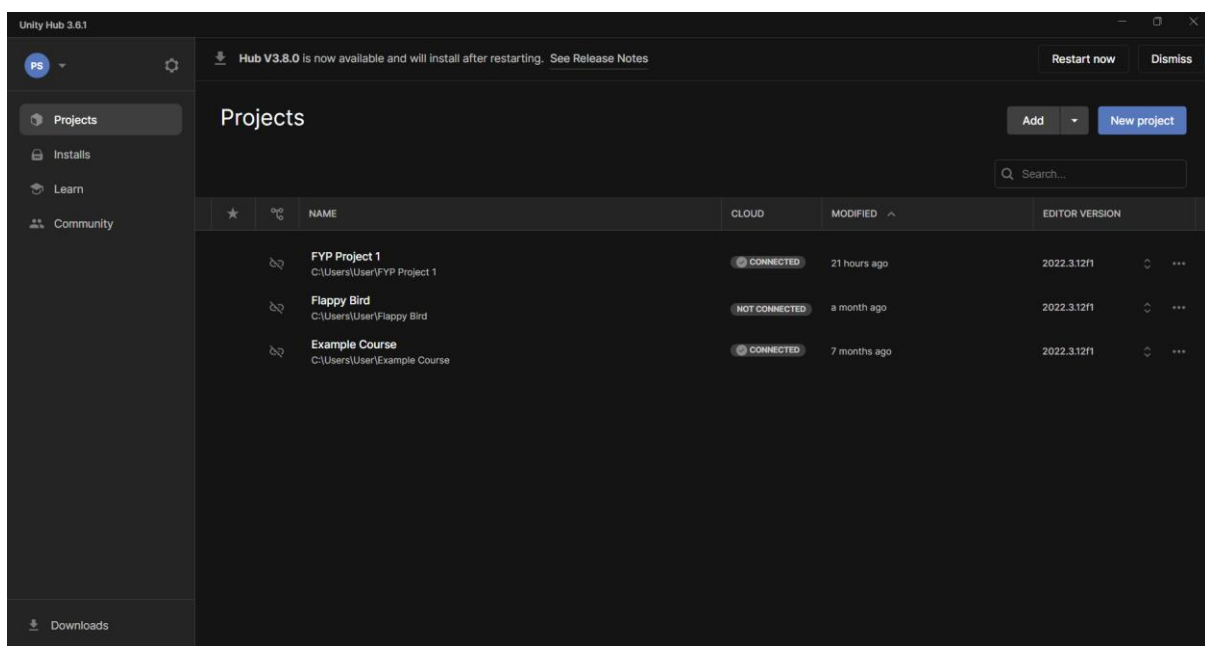


Figure 5.3.1: The Interface of Unity Hub

Next, since we are developing the 2D game, and therefore, select “2D (Buit-in Render Pipeline)” which will automatically do some of the settings for the 2D game by default. After that, wait until the project is created successfully.

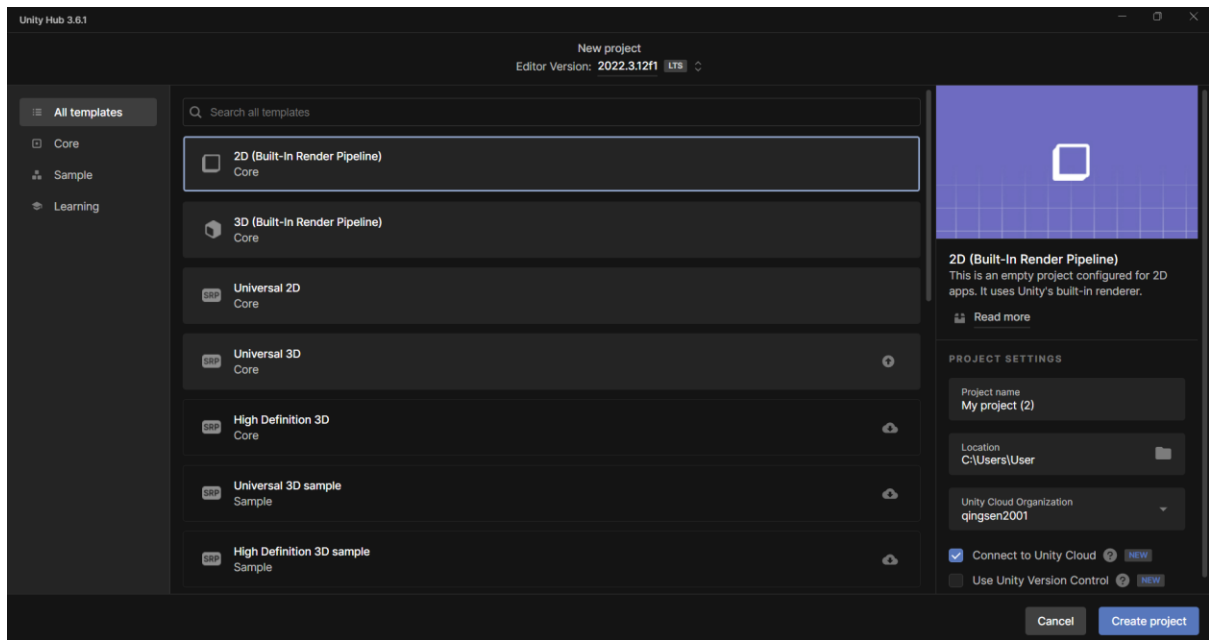


Figure 5.3.2: The new project creation and loading screen

Once the project creation is done, it will have the Unity Editor screen like the figure below.

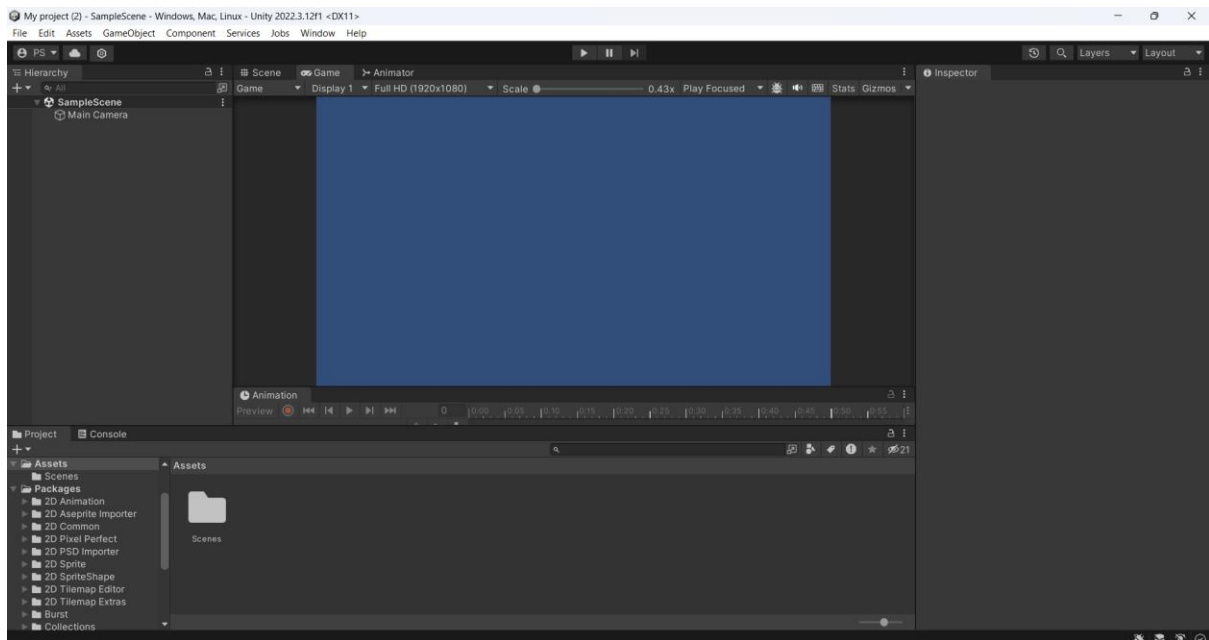


Figure 5.3.3: The Interface of Unity Editor

5.3.2 Create new files

The way to create new files such as new C# scripts, text files, assets or other relevant files is quite simple just right-click on the Project panel below, and select *Create*, then all the file types will be shown as in the figure below.

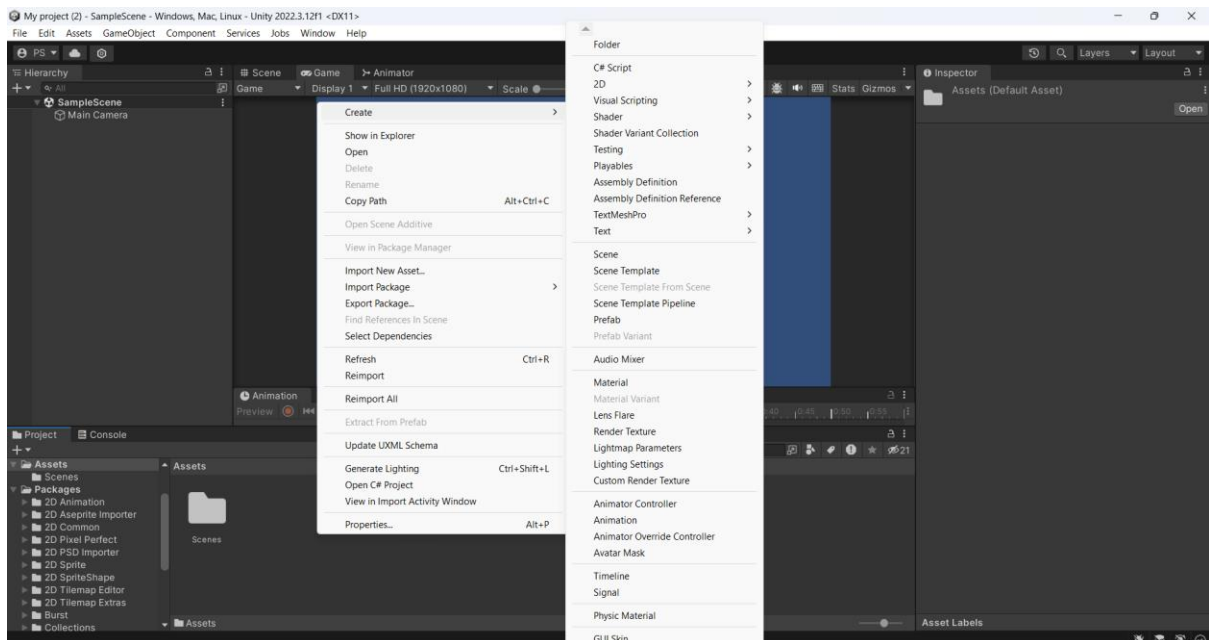


Figure 5.3.4: Create new file in Unity Editor

5.3.3 Create new game objects and its components

Game objects refer to the fundamental objects in Unity that consist of the components, which will act as the behaviours of such objects.

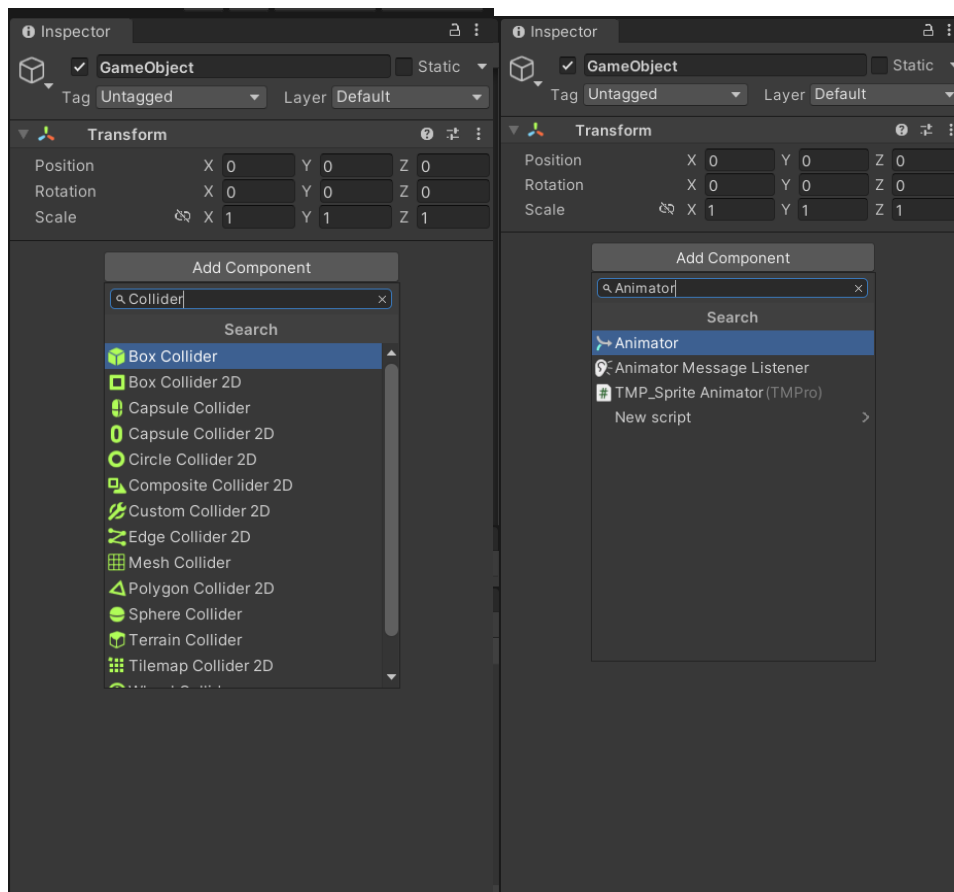
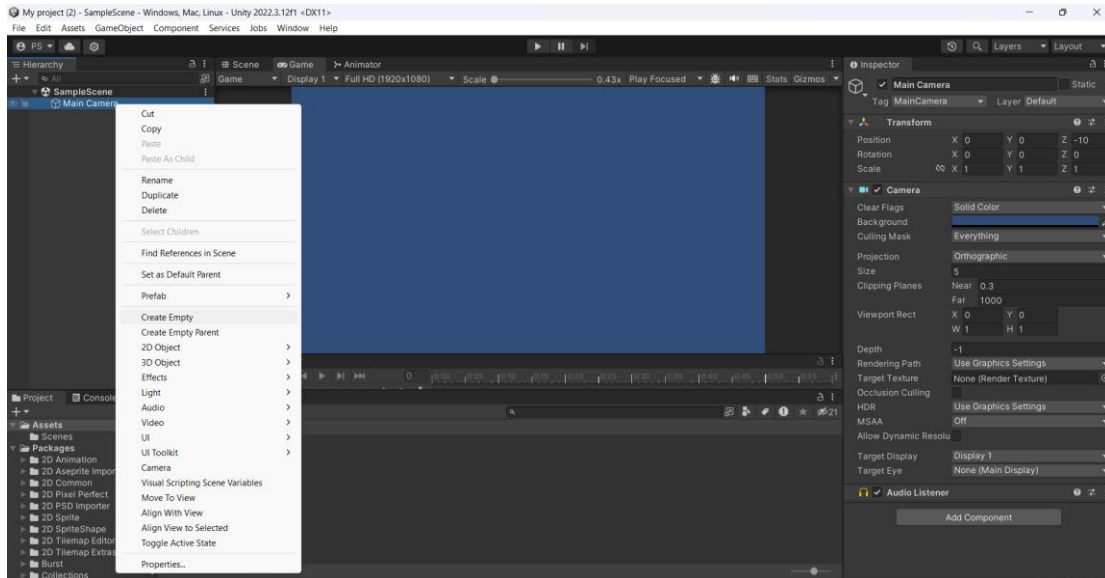


Figure 5.3.5: Create New Game Object and Add New Components in Unity Editor

The way to create new game objects in Unity Editor is also quite direct, which right any current game object of the scene, select *Empty* for empty game objects which does not consist of any default components or select based on the requirement such as *UI* objects for default settings for UI.

5.3.4 Run a developed game in development mode

The game might need to be tested during the development phase, and we can just execute our game if necessary. To run the game, just click on the play button as shown in the figure below. Ensure that the current display is *Game* instead of *Scene*.



Figure 5.3.6: Run the game in development mode

5.3.5 Export to executable file

This step is the final step which needs to be done after all the development and testing are done. To export the game to an executable file, we just select *File* in the left corner, then click on *Build Settings*, we should be able to select the scene needed to build and build the game based on our requirements such as the platform of the game.

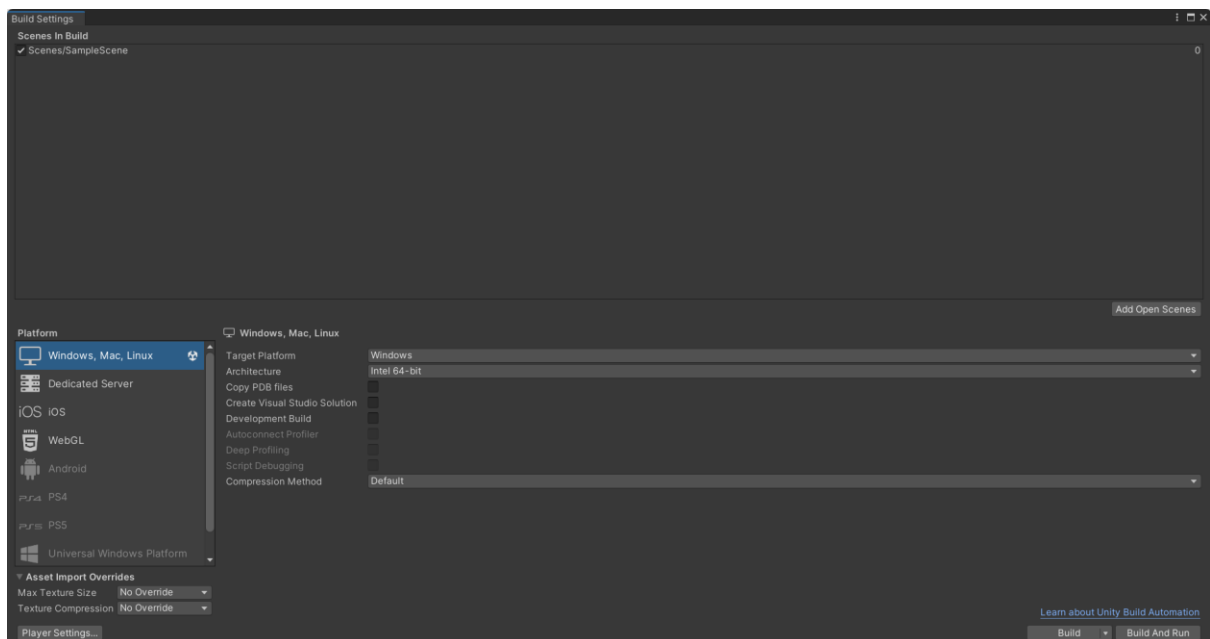
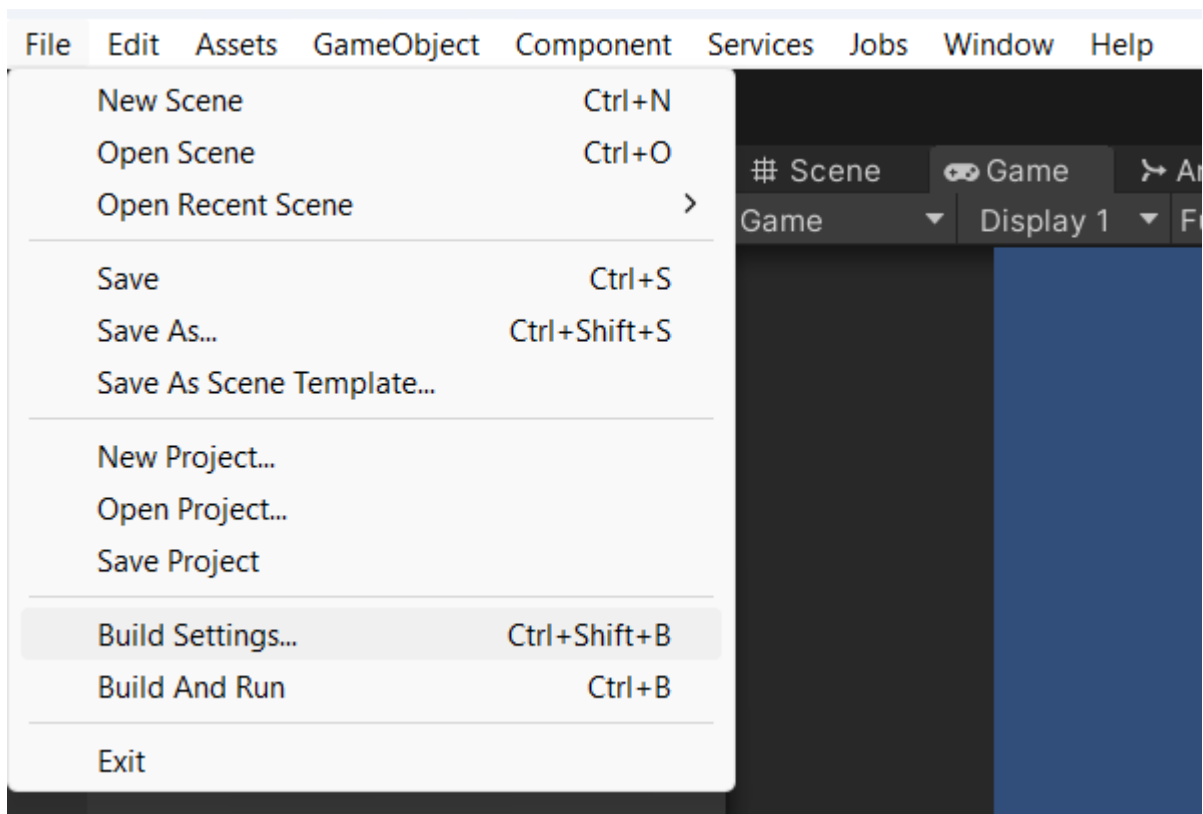



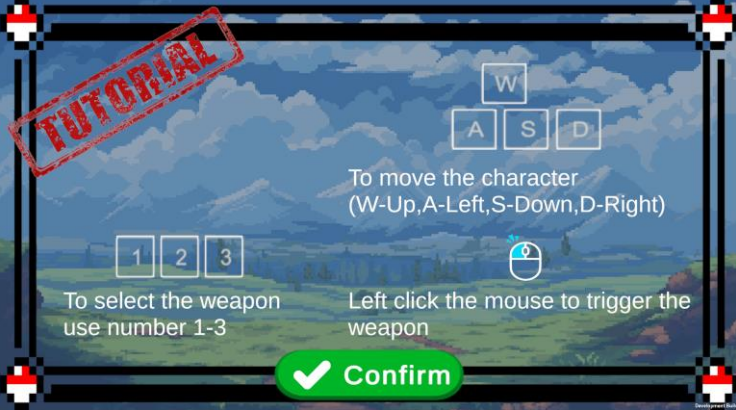
Figure 5.3.7: Build the game and export to executable file

5.4 System Operation (with Screenshot)

System Operation refers to the overall system workflow from the beginning of the system to the end of the system.

5.4.1 Main Menu and Tutorial

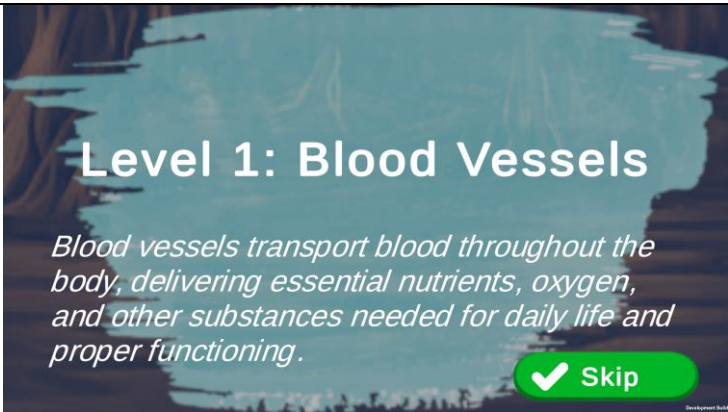
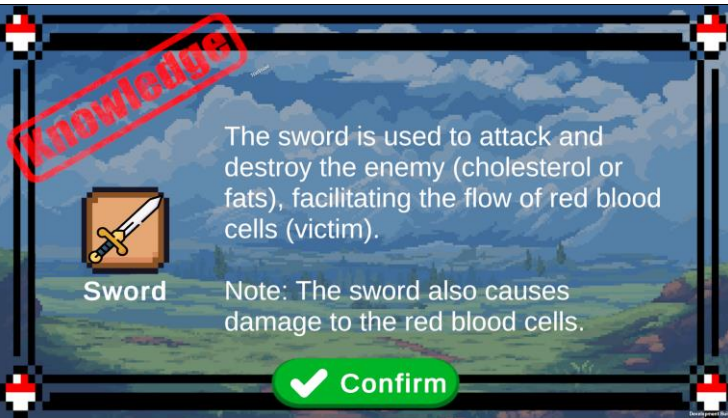
Table 5.4.1: System Operation for the Main Menu and Tutorial

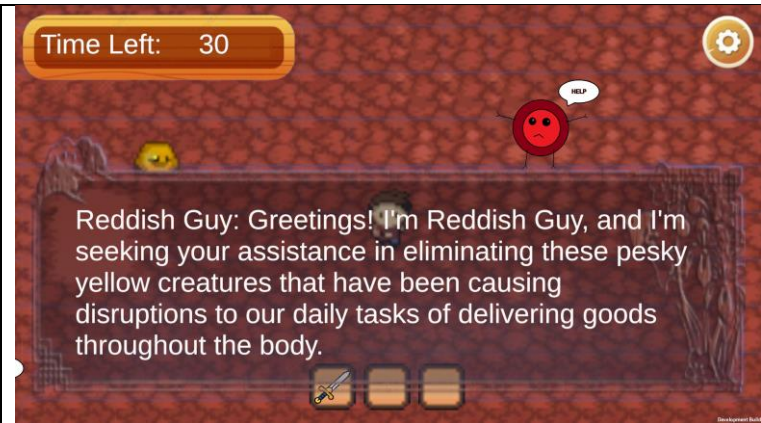
Screenshot	Explanation
	<p>This is the main page where the player first enters the game. The title of the game is “Gamification in learning obesity” and there are two buttons for the player to choose to press; the Play button to start the game and, the exit button to exit the game.</p>
	<p>Once the player chooses to start the game, it would appear the screen where teach the player the basic movement and control for playing the game such as selecting the weapon and moving the character. There is only one button for the player to</p>

	press where to continue with the game.
--	--

5.4.2 Level 1

Table 5.4.2: System Operation for Level 1

Screenshot	Explanation
	<p>After the player chooses the “confirm” button, they will enter the first level of the game which is the blood vessels. There is also the knowledge about the function of the blood vessels shown to the player. There is only one button which allows the player to continue with the game.</p>
	<p>After the player presses the button “Skip”, it will appear the knowledge for the weapon that allowed for player to choose in the first level which is “Sword”. There is only one button also to allow the player to continue with the game.</p>



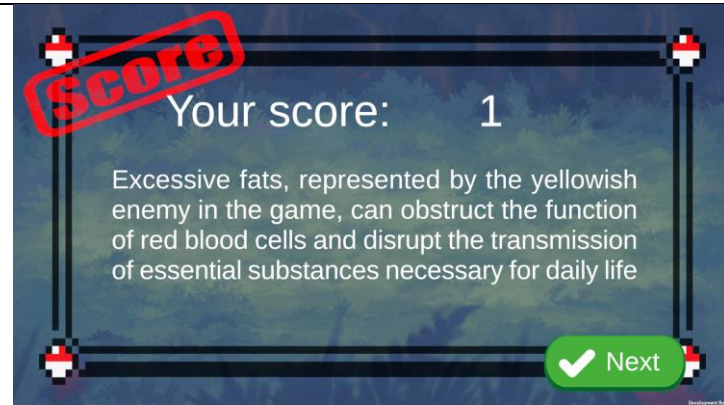
After the player had done the knowledge reading the gameplay for the first level would start. It would first appear in the dialogue between the player and the reddish guy which is the red blood cell (RBC), and after finishing the dialogue, the player will engage in the game. There is one option button that helps the user exit the game or pause the game.



In this level, if the player accidentally hits the RBC, the RBC will change its animation and the score deduction should be recorded. If the player uses the “Sword” to kill the enemy, the score addition should also be recorded, and the enemy would be destroyed after death.




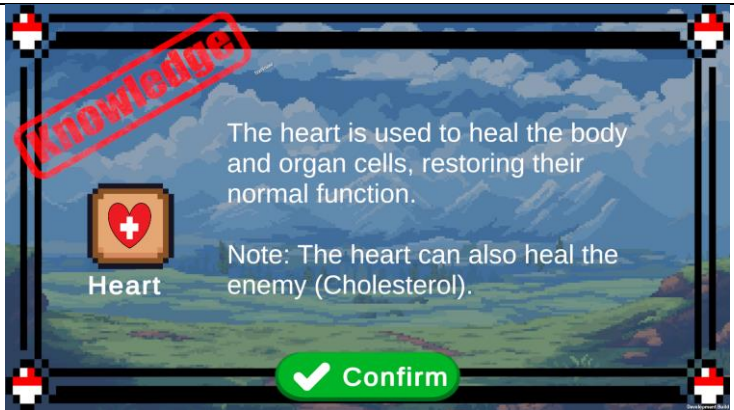
After the time is out, it would enter to the score page which calculate the number of times that the player hit each of the enemies and victims. There is also only one button for the player to continue with the game.

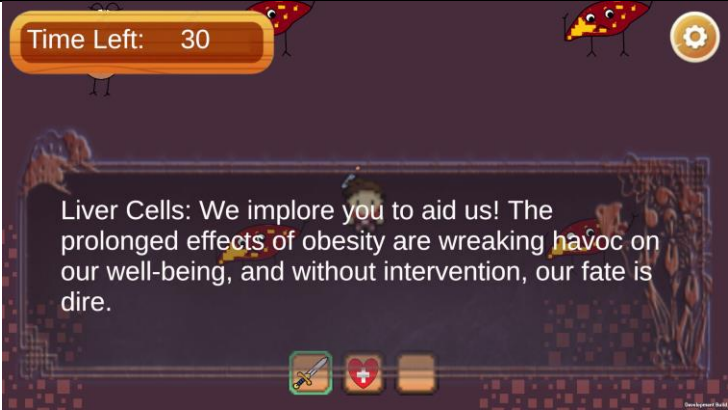



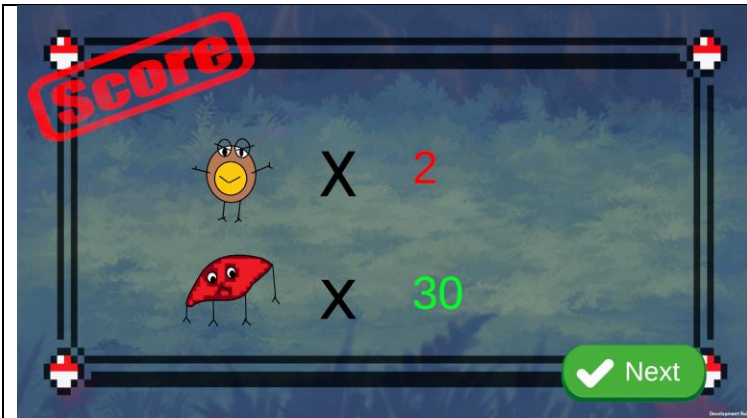
After the player clicks the “Next” button, it calculates the total score obtained by the player. There is also the knowledge to teach the player about the fats toward the blood vessels and one button for the player to continue with the game.

5.4.3 Level 2

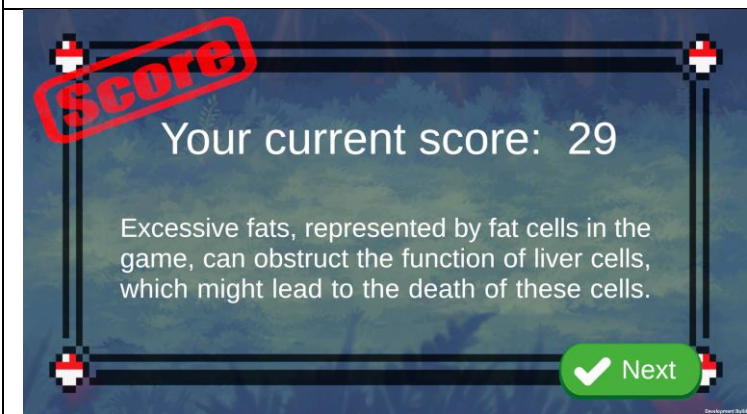
Table 5.4.3: System Operation for Level 2

Screenshot	Explanation
	<p>After the player chooses the “Next” button, they will enter the second level of the game which is the liver cells. There is also the knowledge about the function of the liver cells shown to the player. There is only one button which allows the player to continue with the game.</p>
	<p>Before starting with the level 2 gameplay, there is also knowledge about the new weapon that would be used by the player which is the heart. There is only one “Confirm” button for the player to continue with the game.</p>

	<p>After the player had done with the knowledge for the second level, it would start the game. The dialogue between the player and the liver cells would first appear, and after finishing the dialogue, the player will engage in the game. There is one option button also that helps the user exit the game or pause the game. Note that the heart weapon had been enabled for the player at the current level.</p>
	<p>There are two ways to play the game; One is using the “Sword” to kill the enemy, but it would deduct the score when accidentally hit the victims. Another one is using the “Heart” to heal the victim, but it would deduct the score when accidentally healing the enemy.</p>



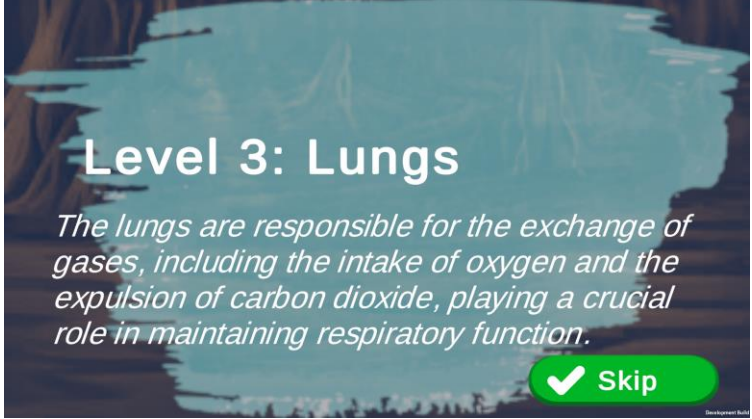
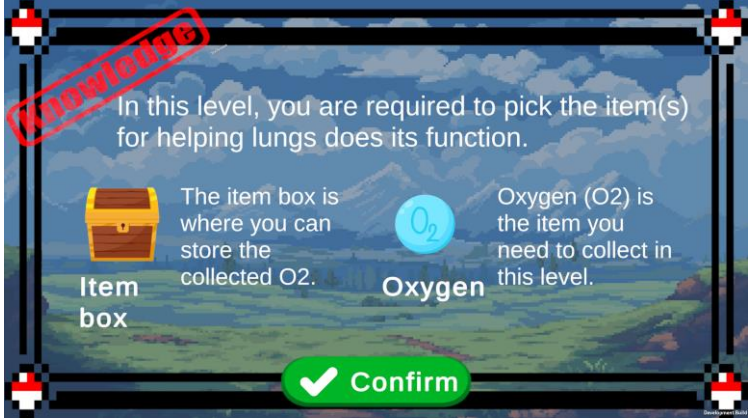
After finishing the counting, the score page for Level 2 will appear. It would show the score obtained based on different characters which includes healing and attacking. There is only one “Next” button for the player to continue with the game.

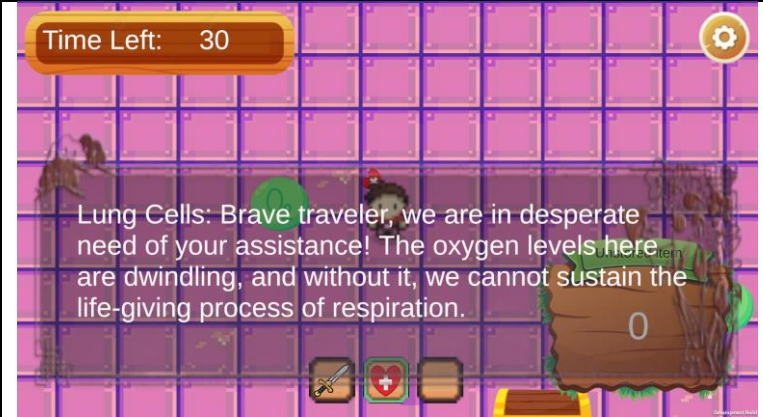



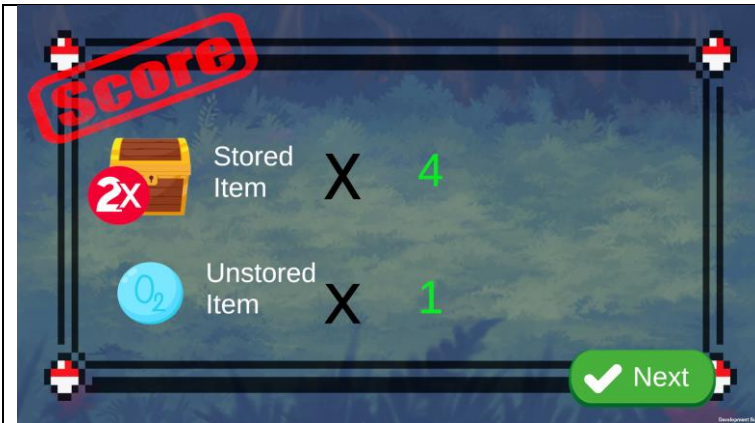
After the player clicks the “Next” button, it calculates the total score obtained by the player in both levels. There is also the knowledge to teach the player about the fats toward the liver cells and one button for the player to continue with the game.

5.4.4 Level 3

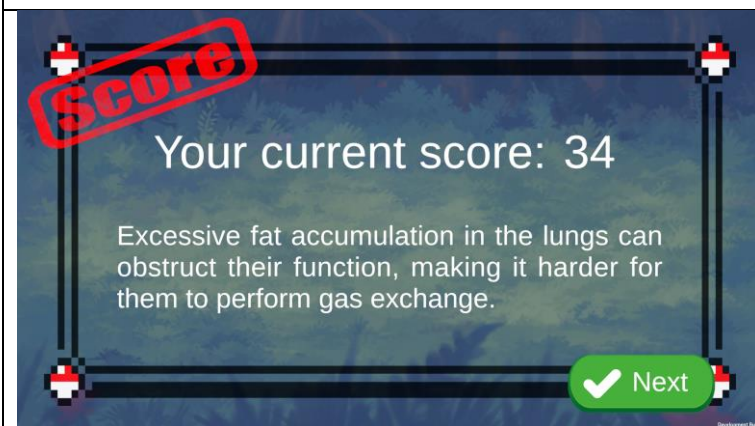
Table 5.4.4: System Operation for Level 3

Screenshot	Explanation
	<p>After the player chooses the “Next” button, it will enter the third level of the game which is lungs. There is also the knowledge about the function of the lungs shown to the player. There is only one button which allows the player to continue with the game.</p>
	<p>Before entering the game, it will show the knowledge about Level 3 which is the item box, and another is the oxygen that needs to be picked up by the player in the game. There is only one button which allows the player to continue with the game.</p>

	<p>After entering Level 3, it would show the dialogue between lung cells and the player. After the dialogue is over, the player can start the gameplay.</p>
	<p>When the player is near the targeted object, the “Pick up” button appears and once the player clicks the button, the oxygen game object will be destroyed, and the scoreboard will record each successful pickup.</p>
	<p>Once the player found the item box and near it, the “Place” button would appear to the player. Once the player placed the collected item, the scoreboard would reset, and the score would calculate respectively.</p>



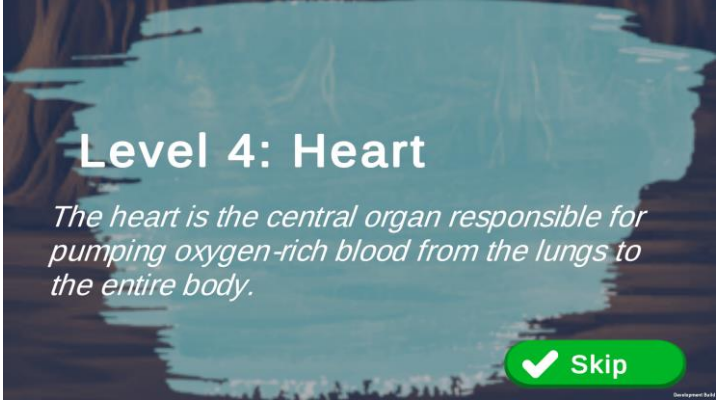
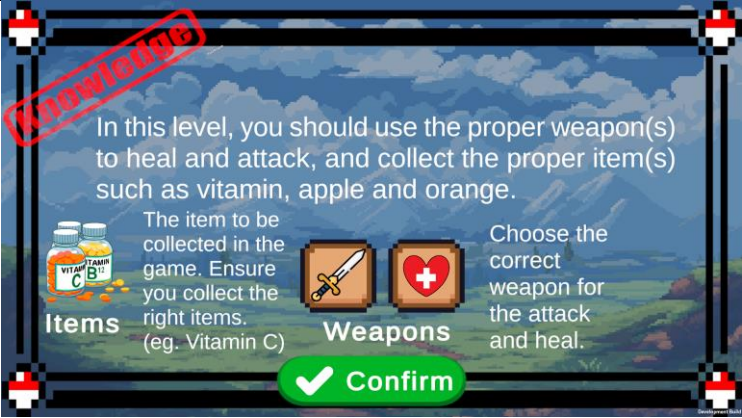
After the time is up, the score page summarises the score for the current level based on the category. There is one next button for the player to continue with the game.



After the player clicks the “Next” button, it calculates the total score for the player so far from level 1 to level 3. Moreover, it would show the knowledge about the fats in the lungs and one next button for the player to continue with the game.

5.4.5 Level 4

Table 5.4.5: System Operation for Level 4

Screenshot	Explanation
	<p>After the player clicks the “Next” button, it will straight away enter Level 4 which is the heart. There is basic knowledge about the function of the heart and one skip button for the player to continue.</p>
	<p>After the player clicks the “Skip” button, it will show the knowledge need to know about Level 4 including the items and weapons. This level combines the gameplay mode for all previous levels. There is also one “Confirm” button for the player to start level 4.</p>



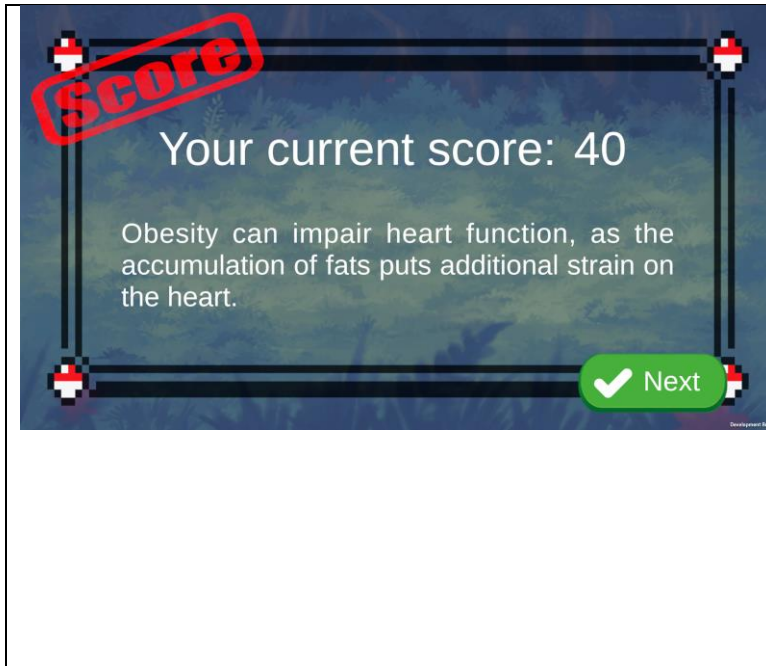
When level 4 is started, the dialogue between the player and the heart cell will appear. The gameplay would start after the dialogue is finished.



The play mode is the same as in previous levels, which is to kill the enemy heal the victim using an appropriate weapon and pick up the correct item that helps with the health of the heart.



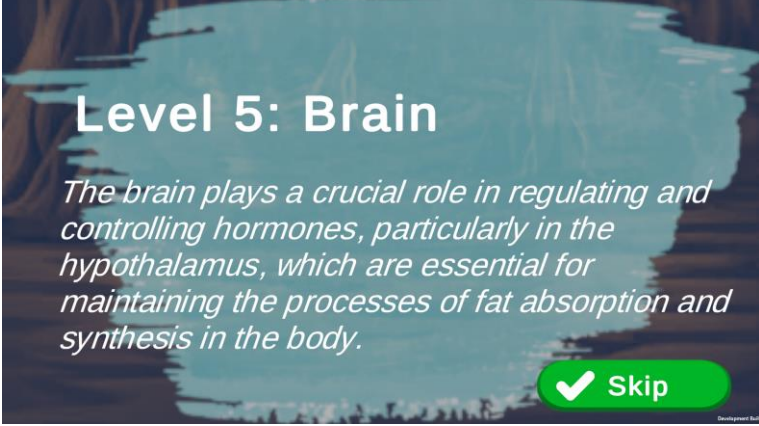
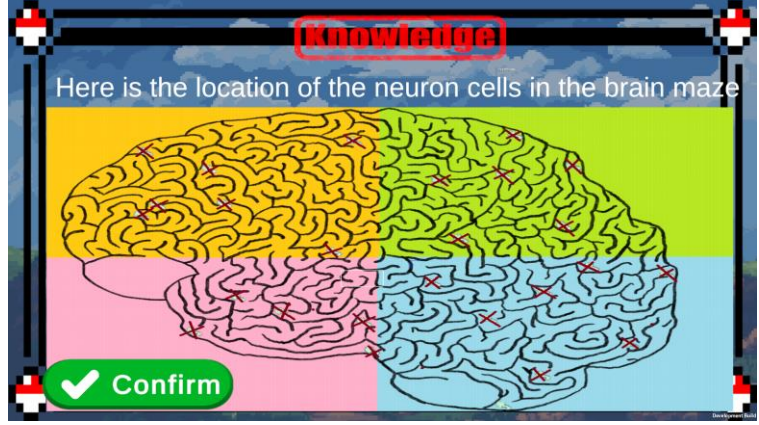
After the time is up, it will summarise the score obtained by the player at the current level according to different characters or categories. There is only one button for the player to continue with the game.

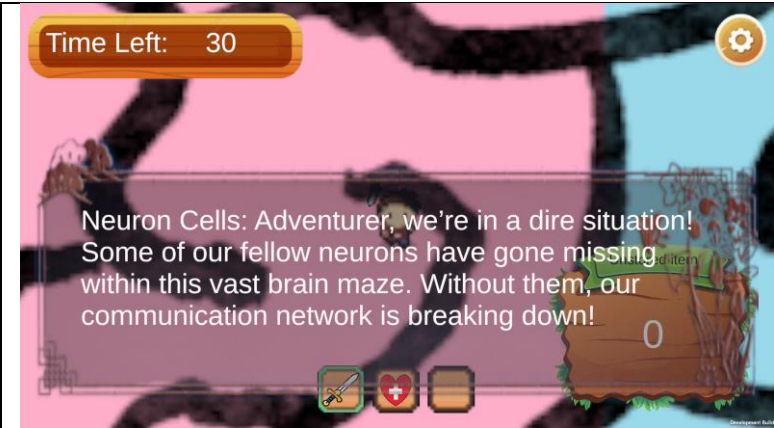


After the player clicks the “Next” button, it calculates the total score for the player so far from level 1 to level 4. Moreover, it would show the knowledge about obesity toward the heart and one next button for the player to continue with the game.

5.4.6 Level 5

Table 5.4.6: System Operation for Level 5

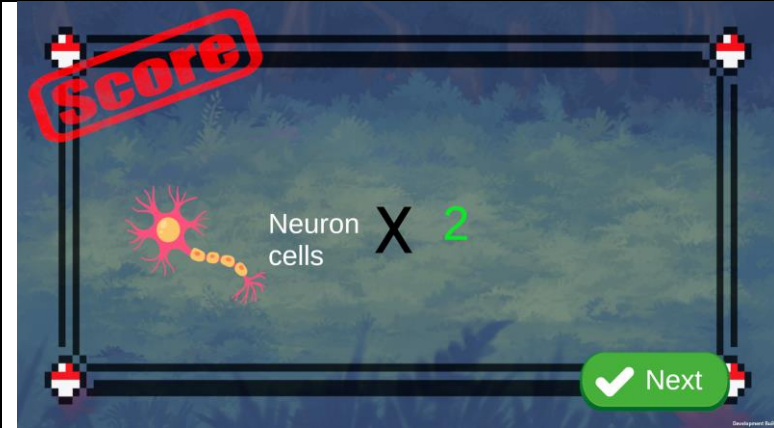
Screenshot	Explanation
	<p>After the player chooses the “Next” button, it will enter the final level of the game which is the brain. There is also the knowledge about the function of the brain shown to the player. There is only one button which allows the player to continue with the game.</p>
	<p>Before starting the gameplay for level 5, the location of all the neuron cells would be shown to the player. The player's objective in this level is to find the neuron cell in this big brain maze. There is one confirm button for the player to start the game.</p>



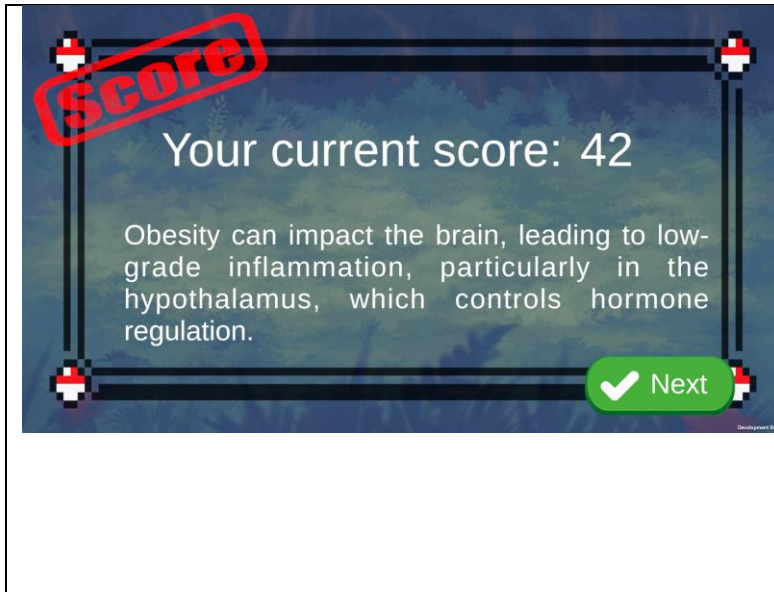
When the player enters Level 5, it first appears the dialogue between the player and neuron cells, the game only starts after the dialogue. The player must move around the maze to find the neuron cells.



When the player is near the neuron cells, the pickup button would appear, and the scoreboard would record the number once the player clicked the button.




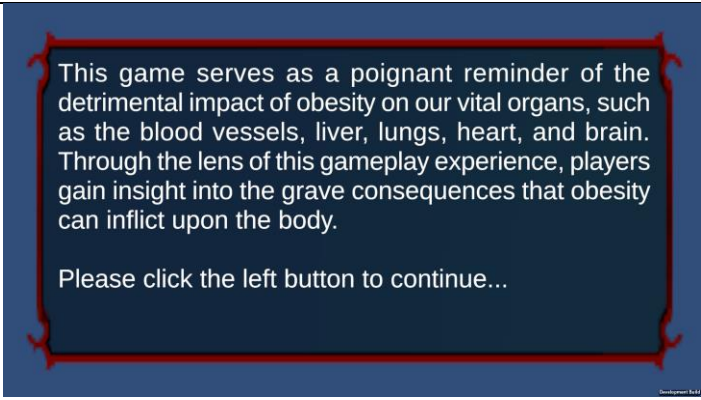
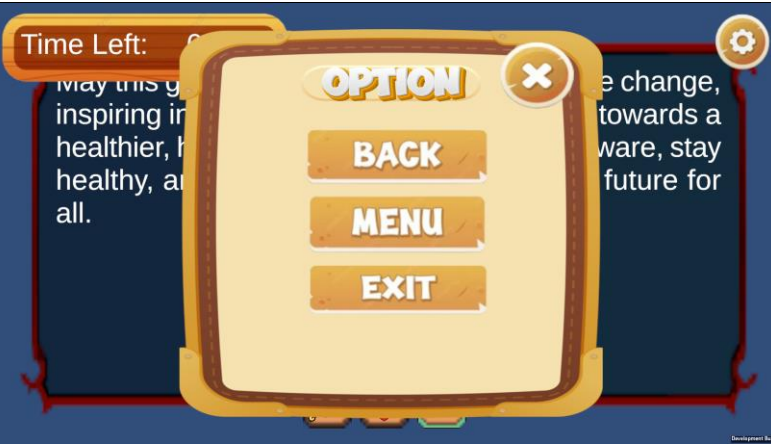
When the time is out, the score page would appear and show how many neuron cells the player had picked up. There is only one button for the player to continue.



After the player clicks the “Next” button, it summarises the score from Level 1 to Level 5 and shows the knowledge about the effect of obesity on the brain. Players can proceed with the game by pressing the next button.

5.4.7 Ending

Table 5.4.7: System Operation for Ending

 <p>The screenshot shows a dark blue background with a decorative border. At the top center, there is a gold medal icon and a red-bordered box containing the word "Rank" in red. Below this, the text "Your Rank: Satisfactory" is displayed in white. Underneath, a paragraph of text reads: "Decent work. You have a basic understanding of how obesity affects the body, but there are several aspects that require further attention." At the bottom right, there is a green button with a white checkmark and the word "Next".</p>	<p>When the player clicks the “Next” button, it shows the ranking for the player based on the total score obtained by the player with its description. The player can proceed with the ending by clicking the next button.</p>
 <p>The screenshot shows a dark blue background with a decorative border. A white text box contains the following text: "This game serves as a poignant reminder of the detrimental impact of obesity on our vital organs, such as the blood vessels, liver, lungs, heart, and brain. Through the lens of this gameplay experience, players gain insight into the grave consequences that obesity can inflict upon the body." Below this, it says "Please click the left button to continue...".</p>	<p>The message box would appear, and the player had to finish the message by left click.</p>
 <p>The screenshot shows a dark blue background with a decorative border. In the foreground, there is a yellow-bordered menu box titled "OPTION" with a close button (X). Inside the menu, there are three buttons: "BACK", "MENU", and "EXIT". To the left of the menu, there is a "Time Left:" label and some partially visible text. To the right, there is a gear icon and some partially visible text.</p>	<p>Once all the message is done, the option menu will be shown to the user and the user can choose to exit the game.</p>

5.5 Implementation Issues and Challenges

One of the implementation issues faced during this project was the limitation of the assets that were available and their suitability. Because of the novelty of our project, the resources such as the graphics for the player, maps and other elements that are suitable for this game are quite rare to find on the internet, even some of them can be found easily in certain website, but it requires payment to use such asset, so it is challenging to implement this game. As a result, there might be delays or overhead in development since additional time is required for the search of the assets or customizing the resources.

Another significant implementation issue to consider is the resource consumption of the computer device. Since the development is running on a local environment, it's crucial to be aware of potential resource constraints such as RAM usage, physical storage, and memory consumption. All the tools, including the unity engine, Unity Hub, Unity Editor, and the integrated development environment (IDE) on a single machine required significant resource utilization. Managing these resources wisely is essential to ensure smooth game development and prevent performance issues.

Furthermore, new fields and language studies during a short period might be challenging. As the game revolves around a novel concept related to the effects of obesity on human organs, there may be a need to explore unfamiliar areas such as medical research, physiological simulations, or educational game design. Navigating these new territories may require a deep understanding of complex subject matter and translating it into engaging gameplay mechanics while ensuring accuracy and relevance to the game's objectives.

Additionally, the design and implementation process itself may pose challenges. Balancing gameplay mechanics with educational content while maintaining player engagement is a delicate task. Design decisions such as level progression, difficulty curve, and user interface design must be carefully considered to create an enjoyable and informative gaming experience. Furthermore, implementing these design choices effectively within the constraints of time, budget, and technical limitations may require iterative testing and refinement throughout the development process.

5.6 Concluding Remark

As a summary, in this chapter, the preparation of the development process, including the registration of the Unity account, guidelines to install the Unity Hub, Unity Editor, and Visual Studio IDE is discussed. In addition, there were also the tips and way to implement the game design with the Unity Editor, including how to add new game object, start the development environment, create new files and add the new components to a game project. Despite that, the system operation which starting from the main menu and tutorial until the ending had been demonstrated in this chapter with screenshot and explanation. Despite encountering various challenges, particularly with asset acquisition, resource management, and balancing gameplay with educational content, the development process made significant progress that bring to a functional and engaging game.

Chapter 6

System Evaluation and Discussion

In this chapter, we will clearly explain the system design for our game application which includes the system block diagram, the system components specifications, the system logic and the integration of knowledge with the game.

6.1 System Testing and Performance Metrics

6.1.1 Functional Testing

The functional testing would verify that all the game features including level progression, score calculation, user interface interaction, dialogue system and collision detection function as designed.

- Level Progression

Level progression refers to the transitions between levels. The testing would be done to ensure smooth transitions, correct loading of assets and scenes, and correct game mechanics for each level.

- Collision Detection

Collision detection refers to the colliders of the game objects which will trigger the game mechanics. This testing would ensure that player colliders do not overlap with walls or obstacles, by leveraging Raycast methods for accurate detection and that the game mechanics can work properly such as the collision between weapon and enemy.

- Score Calculation

This testing would validate the scoring logic and rank assignment throughout the game, which to ensure player receive correct feedback based on their performance in the game.

- UI Interaction

This testing would validate that all the buttons, menus (including options and dialogues), and scoreboards function properly in response to user interactions.

- Dialogue System

This testing verifies that dialogue could be displayed and hidden correctly based on game events.

- Edge Case Testing

This testing covers some of the rare cases that might cause the game to fail to function as expected.

6.1.2 Non-Functional Testing

Non-functional testing refers to the testing done on the elements that may not be necessary for the game to function properly but are required to increase the performance of the game.

- Performance Testing

This testing measures the performance of the game based on the game's responsiveness, load times and frame rate consistency across different levels.

- Compatibility Testing

This testing is done to test the game on different hardware configurations or platforms to ensure that the game can run smoothly on different machines or different environments. In this project, we would try to export this game to other platforms which is WebGL only due to the cost constraints for different hardware.

6.1.3 Performance Metrics

To assess the overall performance of this game, multiple performance metrics were tracked which focus on responsiveness, resource utilization and in-game behaviour.

Key Performance Metrics:

- Frame Rate (FPS)

Frame rate or frame per second (FPS) refers to how many frames would be shown to the user in a second. The game's frame rate is monitored to ensure consistent performance across devices. The targeted FPS for the testing would be 60 FPS.

- Load Times

Load times of the game measured time used during the game's scene transitions to ensure loading or unloading times are within an acceptable range, aiming for under 3 seconds between levels.

- Memory Usage

Memory consumption for a game is important since it is to ensure efficient usage of resources, preventing memory leaks and excessive use that might slow down the game.

- CPU utilization

CPU utilization for a game is crucial since it ensures that the system remains responsive during gameplay. The targeted CPU utilization is around 20-50% for this game.

- Response Time

Response time refers to the time taken for the game which is required to react to user input. It is measured by evaluating the delay between key presses, mouse clicks and the corresponding in-game action.

6.2 Testing Setup and Result

In this section, some of the setup steps before testing and the result of the testing will be discussed in different categories, including functional testing, non-functional testing and the result of performance metrics.

6.2.1 Functional Testing Setup and Result

Before the start of the testing, ensure that all the scene is properly hidden and configured. The testing can be done using the development mode inside the unity or the executable file that is exported to the operating system. Below are the testing results for all functional testing which includes the level progression, collision detection, score calculations, UI interactions, dialogue system and some edge cases testing.

Table 6.2.1: Result of Functional Testing for Main menu and Tutorial

Testing Criteria	Expected Result	Actual Result
Main menu and Tutorial		
Main menu page		
Play button	To next page	Pass
Exit button	Exit the game	Pass
Tutorial page		
Confirm button	To next page	Pass
Tutorial	Correct tutorial	Pass

Table 6.2.2: Result of Functional Testing for Level 1

Testing Criteria	Expected Result	Actual Result
Level 1		
Introduction page		
Skip button	To next page	Pass
Introduction and knowledge	Display correctly	Pass
Knowledge page		

Confirm button	Start Level 1 gameplay	Pass
Knowledge about sword	Display correctly	Pass
Level 1 Gameplay		
Dialogue system	Shows correctly	Pass
Collision between wall and player	Not allow player to bypass the wall	Pass
Collision between sword and enemy	Add score and destroy enemy game object	Pass
Collision between sword and victim	Deduct score and change victim animation	Pass
Player movement	Move as the player input	Pass
Player animation	Change as the movement	Pass
Timer counts down	Count down and end the level correctly	Pass
Option menu	Display and work properly	Pass
Weapon selection and spawn	Weapon change as input	Pass
Camera	Follow character movement	Pass
After Level 1 Gameplay		
Score page		
Victim score	Calculated correctly	Pass
Enemy score	Calculated correctly	Pass
Next button	To next page	Pass
Assets and image	Display correctly	Pass
Total score page		
Total score	Calculated correctly	Pass
Knowledge about first level	Display correctly	Pass
Next button	To next page	Pass

Table 6.2.3: Result of Functional Testing for Level 2

Testing Criteria	Expected Result	Actual Result
Level 2		
Introduction page		
Skip button	To next page	Pass
Introduction and knowledge	Display correctly	Pass
Knowledge page		
Confirm button	Start Level 2 gameplay	Pass
Knowledge about heart	Display correctly	Pass
Level 2 Gameplay		
Dialogue system	Shows correctly	Pass
Collision between wall and player	Not allow player to bypass the wall	Pass
Collision between sword and enemy	Add score and destroy enemy game object	Pass
Collision between sword and victim	Deduct score and change victim animation	Pass
Collision between heart and enemy	Deduct score and enlarge enemy sprite	Pass
Collision between heart and victim	Add score and change victim animation	Pass
Player movement	Move as the player input	Pass
Player animation	Change as the movement	Pass
Timer counts down	Count down and end the level correctly	Pass
Option menu	Display and work properly	Pass
Weapon selection and spawn	Weapon change as input	Pass
Camera	Follow character movement	Pass
After Level 2 Gameplay		
Score page		

Victim score	Calculated correctly	Pass
Enemy score	Calculated correctly	Pass
Next button	To next page	Pass
Assets and image	Display correctly	Pass
Total score page		
Total score	Calculated correctly	Pass
Knowledge about level 2	Display correctly	Pass
Next button	To next page	Pass

Table 6.2.4: Result of Functional Testing for Level 3

Testing Criteria	Expected Result	Actual Result
Level 3		
Introduction page		
Skip button	To next page	Pass
Introduction and knowledge	Display correctly	Pass
Knowledge page		
Confirm button	Start Level 3 gameplay	Pass
Knowledge about gameplay for level 3	Display correctly	Pass
Level 3 Gameplay		
Dialogue system	Shows correctly	Pass
Collision between wall and player	Not allow player to bypass the wall	Pass
Collision between player and item box	Display the place button	Pass
Collision between player and oxygen item	Display the pickup button	Pass
Place button	Reset scoreboard and record the score properly	Pass
Pickup button	Destroy the item object and record the score properly	Pass

Player movement	Move as the player input	Pass
Player animation	Change as the movement	Pass
Timer counts down	Count down and end the level correctly	Pass
Option menu	Display and work properly	Pass
Weapon selection and spawn	Weapon change as input	Pass
Camera	Follow character movement	Pass
Scoreboard	Display picked up item correctly	Pass
After Level 3 Gameplay		
Score page		
Item box score	Calculated correctly	Pass
Item score	Calculated correctly	Pass
Next button	To next page	Pass
Assets and image	Display correctly	Pass
Total score page		
Total score	Calculated correctly	Pass
Knowledge about level 3	Display correctly	Pass
Next button	To next page	Pass

Table 6.2.5: Result of Functional Testing for Level 4

Testing Criteria	Expected Result	Actual Result
Level 4		
Introduction page		
Skip button	To next page	Pass
Introduction and knowledge	Display correctly	Pass
Knowledge page		
Confirm button	Start Level 4 gameplay	Pass

Knowledge about items and weapons	Display correctly	Pass
Level 4 Gameplay		
Dialogue system	Shows correctly	Pass
Collision between wall and player	Not allow player to bypass the wall	Pass
Collision between sword and enemy	Add score and destroy enemy game object	Pass
Collision between sword and victim	Deduct score and change victim animation	Pass
Collision between heart and enemy	Deduct score and enlarge enemy sprite	Pass
Collision between heart and victim	Add score and change victim animation	Pass
Collision between targeted item and player	Display pickup button	Pass
Collision between not targeted item and player	Display pickup button	Pass
Pickup button	Destroy the item and calculate the score according to the item picked up	Pass
Player movement	Move as the player input	Pass
Player animation	Change as the movement	Pass
Timer counts down	Count down and end the level correctly	Pass
Option menu	Display and work properly	Pass
Weapon selection and spawn	Weapon change as input	Pass
Camera	Follow character movement	Pass
After Level 4 Gameplay		
Score page		
Victim score	Calculated correctly	Pass

Enemy score	Calculated correctly	Pass
Item score	Calculated correctly	Pass
Next button	To next page	Pass
Assets and image	Display correctly	Pass
Total score page		
Total score	Calculated correctly	Pass
Knowledge about level 4	Display correctly	Pass
Next button	To next page	Pass

Table 6.2.6: Result of Functional Testing for Level 5

Testing Criteria	Expected Result	Actual Result
Level 5		
Introduction page		
Skip button	To next page	Pass
Introduction and knowledge	Display correctly	Pass
Knowledge page		
Confirm button	Start Level 5 gameplay	Pass
Knowledge about level 5 objective	Display correctly	Pass
Knowledge about location of neurons	Display correctly	Pass
Level 5 Gameplay		
Dialogue system	Shows correctly	Pass
Collision between wall and player	Not allow player to bypass the wall	Pass
Collision between targeted item and player	Display pickup button	Pass
Pickup button	Destroy the item and add the score	Pass
Player movement	Move as the player input	Pass

Player animation	Change as the movement	Pass
Timer counts down	Count down and end the level correctly	Pass
Option menu	Display and work properly	Pass
Weapon selection and spawn	Weapon change as input	Pass
Camera	Follow character movement	Pass
Scoreboard	Display picked up item correctly	Pass
After Level 5 Gameplay		
Score page		
Neuron score	Calculated correctly	Pass
Next button	To next page	Pass
Assets and image	Display correctly	Pass
Total score page		
Total score	Calculated correctly	Pass
Knowledge about level 5	Display correctly	Pass
Next button	To next page	Pass

Table 6.2.7: Result of Functional Testing for Ending

Testing Criteria	Expected Result	Actual Result
Ending		
Ranking page		
Next button	To ending	Pass
Ranking	Display correct ranking according to total score obtained	Pass
Description	Display the correct description for obtained ranking	Pass
Ending		
Message	Display correctly	Pass

Option menu	Display after end of message	Pass
Option menu-Exit button	Exit the game	Pass
Option menu-Close/Back button	Close the option menu	Pass

Table 6.2.8: Edge Case Testing Result

Testing Criteria	Result
The player pauses during important events (e.g., collision, dialogue).	The game pauses and continues after resuming. (Pass)
Two inputs (e.g., weapon selection and movement) are given simultaneously.	The input was received and responded to simultaneously. (Pass)
The timer expires mid-event (e.g., during a sword-enemy collision).	The response works concerning the last collision. (Pass)
Negative Testing (invalid input or actions such as trying to pick up an already collected item, or selecting a disabled button)	Fixed by error handling, such as destroying the item after picking up or hiding the disabled button. (Pass)

6.2.2 Non-Functional Testing Setup and Result

Before starting with the non-functional testing, it is important to export the program by development build and check the option of auto-connect profiler before building the program.

The example can be seen in the figure below:

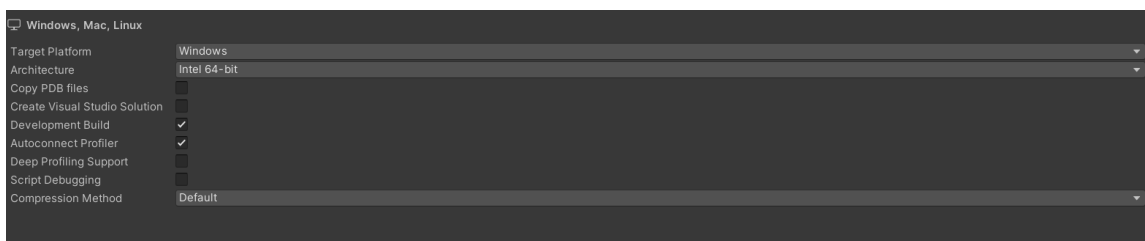


Figure 6.2.1: Development build and auto-connect profiler

However, if we directly test the game inside the unity editor which is the development environment, this step could be ignored. To open the profiler in the unity editor, click the Windows > Analysis > Profiler, and we should see the profiler like in figure 6.2.3.

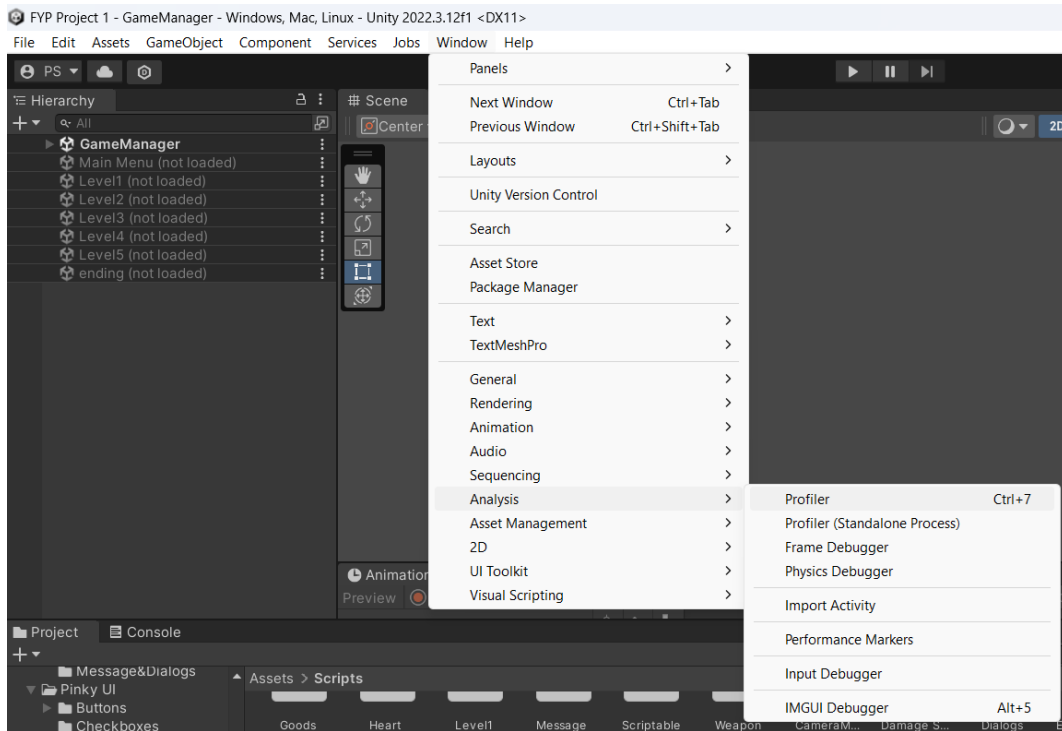


Figure 6.2.2: Open the profiler in Unity Editor

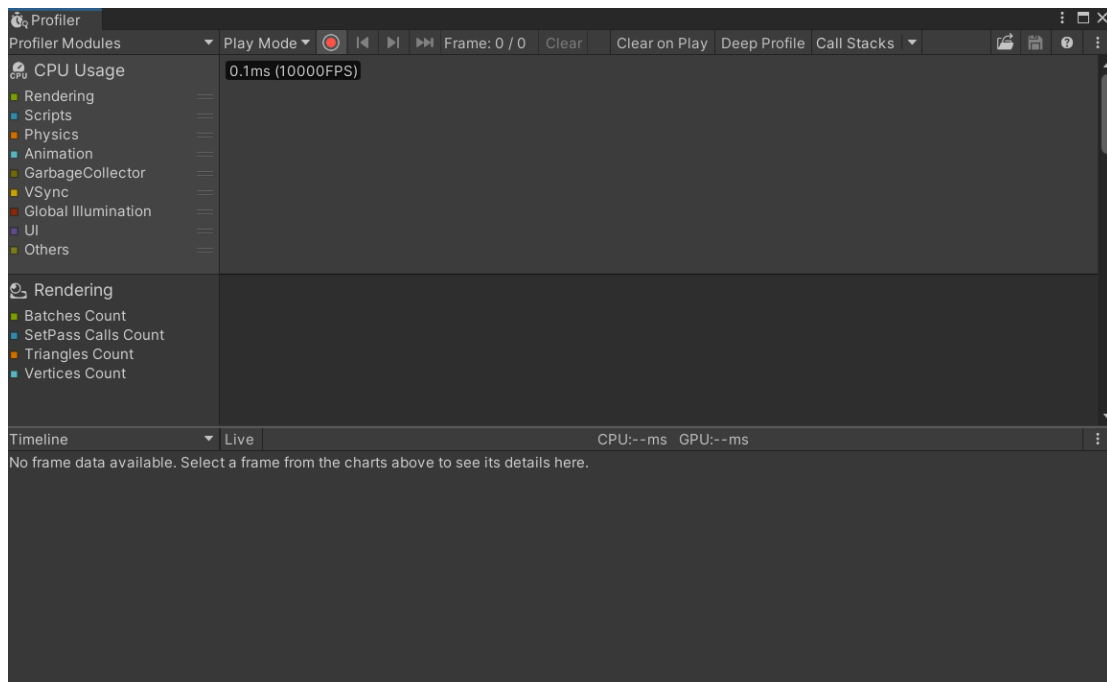


Figure 6.2.3: Profiler in Unity Editor

The profiler would record all the necessary information including the FPS, CPU utilization, response time, load times, and memory usage for us automatically when the game is executed.

Table 6.2.9: Non-functional Testing Result

Testing Criteria	Expected Result	Actual Result
Transition between pages	FPS: above 60 fps CPU: around 20-50% Memory: below 2 GB Load times: below 3s Response: below 3s	FPS: 258 CPU: 23% Memory: 1.5GB Load times: 0.2ms Response: 0.031ms
During gameplay Level 1		FPS: 201 CPU: 32% Memory: 1.5GB Load times: 0.45ms Response: 1.53ms
During gameplay Level 2		FPS: 223 CPU: 31% Memory: 1.5GB Load times: 0.12ms Response: 0.313ms
During gameplay Level 3		FPS: 235 CPU: 33% Memory: 1.5GB Load times: 0.2ms Response: 0.31ms
During gameplay Level 4		FPS: 221 CPU: 32% Memory: 1.5GB Load times: 0.22ms Response: 1.31ms
During gameplay Level 5		FPS: 207 CPU: 31% Memory: 1.5GB Load times: 0.244ms Response: 1.31ms

Compatibility Testing

Before carrying out the compatibility testing, ensure that the build settings have been changed to the WebGL platform for testing in WebGL.

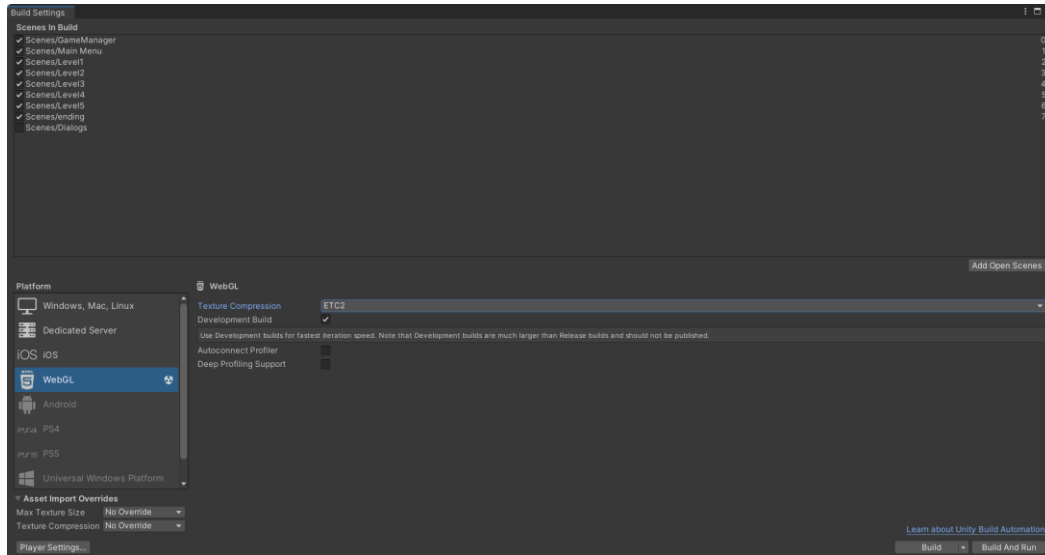


Figure 6.2.4: Build settings for WebGL

Table 6.2.10: Compatibility Testing Result

Testing Platform	Expected Result	Actual Result
Windows 11	Work properly	Pass
WebGL	Work properly	Pass

6.2.3 Result Discussion

Based on the results of the functional testing and non-functional testing in the previous section, the game application passed the testing and performed as expected. The functional testing covered the level progression, collision detection, score calculations, UI interactions, dialogue system and some edge cases testing. In the edge cases testing, some race cases have been tested and the actual result of the testing is working as expected which is considered passed the testing. The game application also passed the non-functional testing including performance testing and compatibility testing. In the performance testing, some of key metrics were used to measure the performance of the game including FPS, CPU utilization, memory utilization, load times, and response times. All the key metrics fall in the acceptable range of the expected result, and therefore, it passed the performance testing. Moreover, in compatibility testing, the game is exported to different platforms, such as Windows 11 and WebGL. The game can work perfectly in both environments and as expected which indicates that the game passed the compatibility testing.

6.3 Project Challenges

Although the project was evaluated and deemed successful, it encountered several challenges during the development and implementation phases. These challenges included, but were not limited to, hardware limitations, time constraints, professional knowledge gaps, and difficulties in acquiring suitable assets or resources.

One of the significant challenges was the hardware limitations posed by the computer devices used for game development and testing. As discussed in Chapter 3, the project was developed using a laptop with specific hardware configurations. While the laptop met the minimum requirements for game development, its limitations in terms of processing power, memory, and

graphics capabilities could potentially impact both the development process and the game's performance. Developing games on such devices might lead to slower performance, prolonged rendering times, or other performance bottlenecks that could delay production.

Another challenge was the time constraint imposed on the project. The development timeline, which included the IIPSPW stage (for initial idea drafting), Project 1 (which assessed the feasibility of the project), and Project 2 (which involved implementation and testing), spanned approximately one year. This relatively short timeframe was limiting, as each phase of the project—preliminary study, idea brainstorming, and the acquisition of the necessary implementation knowledge—required significant time and effort. The tight schedule left little room for unexpected setbacks, refinements, or prolonged iterations during the development process.

The project also faced challenges related to professional knowledge. Game development is an industry characterized by rapid technological advancements and frequent software updates. This meant that the knowledge required for the project had to be continuously updated, as older versions of development tools may no longer be supported. Keeping pace with new software versions, techniques, and methodologies required additional learning, which added another layer of complexity to the project.

Lastly, acquiring assets and resources posed a challenge due to the novel nature of the project. Since the game concept was unique, finding appropriate sprites for the victims and enemies was difficult. While some assets were available online, they were either limited or required customization to fit the project's requirements. This customization added extra work to the

development process, further complicating asset management and increasing the overall workload.

In conclusion, the challenges faced during the project stemmed from hardware limitations, time constraints, gaps in professional knowledge, and difficulties in acquiring or customizing assets. These factors collectively made the development process more challenging but ultimately contributed to the project's growth and success.

6.4 Objectives Evaluation

Throughout the project, we have successfully identified suitable game elements that enhance the efficiency of learning by reviewing several studies, as discussed in Chapter 2. These elements were chosen based on their ability to make the educational process more engaging through interactive gameplay, which aligns with the first objective of this project. Given that we have carefully selected game elements to achieve this goal, we can conclude that the first objective has been fulfilled.

The second objective was to design and develop a 2D game consisting of five levels, using the identified game elements to encourage individuals to learn about the negative effects of obesity on human organs through immersive gameplay. Since we have successfully developed a functional 2D game that incorporates these educational elements, and the knowledge is applied throughout the game's design, we can confidently state that this objective has also been achieved.

The third objective focus on the how to raise public awareness about the effects of obesity and encourage action toward maintaining a healthy lifestyle. Even verifying the success of this objective may require deeper study, within the scope of this project, we addressed it by implementing a ranking system at the end of the game. This ranking is based on the player's score, which reflects how well they understood and interacted with the educational content throughout the levels. By evaluating player performance through this system, we can infer whether they grasped or learned the knowledge provided and understood the consequences of obesity. Therefore, we can reasonably judge that this objective has been partially fulfilled within the scope of the project's design.

6.5 Concluding Remark

In summary, this chapter discussed the evaluation of the system by carried out various testing strategies, functional testing and non-functional testing. Functional testing includes testing on level progression, collision detection, score calculations, UI interactions, dialogue system and some edge cases testing while non-functional testing includes the performance testing and compatibility testing. The testing result for both functional and non-functional testing are passed, which working as expected. This indicates that our game application can functioning properly as expected. Moreover, we have discussed some of the project challenges and evaluate our project objectives in this chapter.

Chapter 7

Conclusion and Recommendation

7.1 Conclusion

This project began with the brainstorming phase in the IIPSPW and has progressed to the current stage of implementation and testing. The main objectives of the project were to identify suitable game elements, build a functional 2D game system, and raise awareness about the effects of obesity on the human body. Through a preliminary study and review of relevant literature, we explored the effectiveness of gamification in learning. Various papers were reviewed, which involved gamification in education, fitness, well-being, and healthcare. These studies helped guide our project, particularly by highlighting the negative impacts of obesity on the body's organs and the contributing factors.

Within our project scope, we focused on the effects of obesity on organs and implemented various game elements, including level progression, a scoring and reward system, and engaging 2D gameplay. Chapter 3 outlined the proposed methodology, which employed the agile development process. This process guided the project from software specifications to software evaluation. The game's design, including its levels, UI, dialogue boxes, and logic, was discussed in detail.

Chapter 5 detailed the system's implementation, covering the registration of a Unity ID, installation of Unity Hub and Editor, Microsoft Visual Studio IDE, and tips for game implementation. The system operation was demonstrated with screenshots and explanations. The project was evaluated using both functional and non-functional testing, covering areas such as level progression, UI interaction, collision detection, the dialogue system, score

calculation, edge case testing, performance testing, and compatibility testing. The testing results indicate that the project passed these assessments, though certain challenges were encountered along the way. Additionally, the project's objectives were evaluated: the first two objectives were fully met, while the third objective was partially achieved through the implementation of a ranking system designed to assess players' understanding of the game's educational content.

In conclusion, this project progressed from the idea generation phase during IIPSPW, through the feasibility studies in Project 1, to the implementation and testing phase in Project 2, resulting in the creation of a fully functional game system aligned with the project objectives.

7.2 Recommendation

While this project is considered successful and has fulfilled its objectives, there are several recommendations for future development. One major area for improvement is the integration of emerging technologies, such as augmented reality (AR) or virtual reality (VR), which could enhance player engagement by enabling physical activity within the game, rather than limiting gameplay to keyboard input on a computer. Another recommendation is to expand the scope of non-functional testing, particularly usability testing involving real users. Conducting surveys or questionnaires would provide valuable feedback on how the game system could be improved and how effectively the knowledge is delivered to players. This would offer insight into potential areas for refinement in future iterations.

Lastly, the game could be expanded to more platforms beyond a standalone PC game, including web-based, mobile, or even console versions. This would allow the game to reach a broader audience, making it accessible to a wider variety of players and not just those using a PC.

REFERENCES

- [1] C. C. Tat, L. W. Kent, S. M. Sallehuddin and S. S. Ganapathy, "Prevalence of overweight and its associated factors among Malaysian adults: findings from a nationally representative survey," *medRxiv*, 2023.
- [2] Y. S. Chin, F. C. Woon and Y. M. Chan, "The impact of Movement Control Order during the COVID-19 pandemic on lifestyle behaviours and body weight changes: Findings from the MyNutriLifeCOVID-19 online survey," *PLOS ONE*, vol. 17, pp. 1-16, 01 2022.
- [3] K. M. MLA. Kapp, *The Gamification of Learning and Instruction : Game-Based Methods and Strategies for Training and Education*, San Francisco, CA :Pfeiffer, 2012.
- [4] D. R. Jamaludin, "Using technology in the classroom can transform learning," *New Straits Times*, 11 June 2018. [Online]. Available: <https://www.nst.com.my/opinion/letters/2018/06/378673/using-technology-classroom-can-transform-learning>. [Accessed Sep 2023].
- [5] R. W. Mee Mee, T. S. T. Shahdan, M. R. Ismail, K. A. Ghani, L. S. Pek, W. Y. Von, A. Woo and Y. S. Rao, "Role of Gamification in Classroom Teaching: Pre-Service Teachers' View," *International Journal of Evaluation and Research in Education*, vol. 9, no. 3, pp. 684-690, Sep 2020.
- [6] J. Krath, L. Schürmann and H. F. v. Korflesch, "Revealing the theoretical basis of gamification: A systematic review and analysis of theory in research on gamification, serious games and game-based learning.," *Computers in Human Behavior*, vol. 125, no. 106963, 2021.
- [7] R. N. Landers, "Developing a Theory of Gamified Learning: Linking Serious Games and Gamification of Learning," *Simulation & Gaming*, vol. 45(6), p. 752–768.
- [8] D. Johnso, S. Deterding, K.-A. Kuhn, A. Staneva, S. Stoyanov and L. Hides, "Gamification for health and wellbeing: A systematic review of the literature,," vol. 6, pp. 89-106, 2016.

- [9] M. Hall, S. O. Kimbrough, C. Haas, C. Weinhardt and S. Caton, "Towards the gamification of well-being measures," *IEEE 8th International Conference on E-Science*, pp. 1-8, 2012.
- [10] Mulcahy, R. & Zainuddin, Nadia, Russell-Bennett and Rebekah, "Transformative value and the role of involvement in gamification and serious games for well-being," *Journal of Service Management*, 2020.
- [11] N. A. Borghese, R. Mainetti, M. Pirovano and P. L. Lanzi, "An intelligent game engine for the at-home rehabilitation of stroke patients," *2013 IEEE 2nd International Conference on Serious Games and Applications for Health (SeGAH)*, pp. 1-8, 2013.
- [12] BRADLEY University, "HOW GAMIFICATION CAN BE USED IN COUNSELING," BRADLEY University, [Online]. Available: <https://onlinedegrees.bradley.edu/blog/how-gamification-can-be-used-in-counseling/>. [Accessed April 2024].
- [13] Merry, S. N, Stasiak, Karolina, Shepherd, Matthew, Frampton, Chris, Fleming, Theresa, Lucassen and M. F. G, "The effectiveness of SPARX, a computerised self help intervention for adolescents seeking help for depression: randomised controlled non-inferiority trial," *BMJ*, vol. 344, 2012.
- [14] S. Jiang, W. Lu, X. Zong, Ruan, H. and Y. Liu, "Obesity and hypertension (Review)," *Experimental and Therapeutic Medicine*, Vols. 2395-2399, p. 12, 2016.
- [15] M. JL, T. HB and S. GB, "Relationship between hepatic morphology and clinical and biochemical findings in morbidly obese patients," *J Clin Pathol*, vol. 26(10), pp. 776-83, Oct 1973.
- [16] Montani, Jean-Pierre, Carroll, J. & Dwyer, T. & Antic, V. & Yang, Z. & Dulloo and AG., "Ectopic fat storage in heart, blood vessels and kidneys in the pathogenesis of cardiovascular diseases," *International journal of obesity and related metabolic disorders: journal of the International Association for the Study of Obesity*, vol. 28 Suppl 4, pp. S58-65, 2005.

- [17] P. Poirier, T. D. Giles, G. A. Bray, Y. Hong, J. S. Stern, F. X. Pi-Sunyer and R. H. Eckel, "Obesity and Cardiovascular Disease: Pathophysiology, Evaluation, and Effect of Weight Loss," *Circulation*, vol. 113, no. 6, pp. 898-918, 2006.
- [18] F. Cazettes, J. I. Cohen, P. L. Yau, H. Talbot and A. Convit, "Obesity-mediated inflammation may damage the brain circuit that regulates food intake," *Brain Research*, vol. 1373, pp. 101-109, 2011.
- [19] C. T. De Souza, E. P. Araujo, S. Bordin, R. Ashimine, R. L. Zollner, A. C. Boschero, M. J. A. Saad and L. A. Velloso, "Consumption of a Fat-Rich Diet Activates a Proinflammatory Response and Induces Insulin Resistance in the Hypothalamus," *Endocrinology*, vol. 146, no. 10, pp. 4192-4199, 2005.
- [20] O. JM and G. CK., "Macrophages, inflammation, and insulin resistance," *Annu Rev Physiol*, vol. 46, pp. 7-219, 2010.
- [21] D. Costa, M. C. Barbalho, G. P. S. Miguel and E. M. P. F. J. L. M. C. Azevedo, "The impact of obesity on pulmonary function in adult women," *Clinics*, vol. 63, no. 6, pp. 719-724, 2008.
- [22] Health news, "Playing video games linked to higher BMI in kids," University Of Leeds, 6 Apr 2020. [Online]. Available: <https://www.leeds.ac.uk/news-health/news/article/4571/playing-video-games-linked-to-higher-bmi-in-kids#:~:text=They%20found%20that%20children%20who,to%2C%20such%20as%20watching%20television.> [Accessed April 2024].
- [23] Harvard Health Publishing, "11 foods that lower cholesterol", Harvard Health, 2024. Available: <https://www.health.harvard.edu/heart-health/11-foods-that-lower-cholesterol> (Accessed: April 2024).
- [24] Johns Hopkins Medicine, "Nonalcoholic fatty liver disease", Johns Hopkins Medicine, 2019. Available: <https://www.hopkinsmedicine.org/health/conditions-and-diseases/nonalcoholic-fatty-liver-disease> (Accessed: April 2024).

APPENDIX

Level Manager Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using TMPro;

public class LevelManager : MonoBehaviour
{
    public static LevelManager Instance { get; private set; }
    public GameObject playerPrefab;
    private bool isdialog = true;
    public Level level1 = new Level();
    public Level level2 = new Level();
    public Level level3 = new Level();
    public Level level4 = new Level();
    public Level level5 = new Level();
    public GameObject optionCanvas;
    [SerializeField] public TextMeshProUGUI timerText;
    [SerializeField] public float duration = 30;
    [SerializeField] private GameObject eventSystem;
    private float timer;
    public string currentLevel;
    public GameObject pickupbtn;
    public GameObject inventory;
    public GameObject heartInventory;
    public GameObject dropbtn;
    public GameObject board;
    public TextMeshProUGUI boardscore;

    private int currentScore = 0;

    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }

        GameObject temp = eventSystem.transform.Find("UICanvas").gameObject;
        pickupbtn = temp.transform.Find("Pickupbtn").gameObject;
        dropbtn = temp.transform.Find("Dropbtn").gameObject;
        inventory = temp.transform.Find("ActiveInventory").gameObject;
        heartInventory = inventory.transform.Find("Inventory1").gameObject;
        board = temp.transform.Find("board").gameObject;
        boardscore = board.transform.Find("number").GetComponent<TextMeshProUGUI>();
    }

    public void Start() //Start the first scene
    {
        /*
        setEvent(true);
        SceneManager.LoadSceneAsync("Level5", LoadSceneMode.Additive);
        */
    }
}
```

```

    level5.SetActive(true);
    currentLevel = "Level5";
    timerText.text = duration.ToString("F0");
    timer = duration;
    */

    setEvent(true);
    SceneManager.LoadSceneAsync("Level1", LoadSceneMode.Additive);
    level1.SetActive(true);
    currentLevel = "Level1";
    timerText.text = duration.ToString("F0");
    timer = duration;
}

public void Update()
{
    timerText.text = timer.ToString("F0");
    if (level1.getActive())
    {
        if (timer > 0)
        {
            timer -= Time.deltaTime; //Update timer
            if (timer <= 0)
            {
                level1.setPass(true);
            }
        }
        if (IsLevelPassed(level1))
        {
            level1.SetActive(false);
            ChangeToNextLevel();
        }
    }

    else if (level2.getActive())
    {
        if (timer > 0)
        {
            timer -= Time.deltaTime; //Update timer
            if (timer <= 0)
            {
                level2.setPass(true);
            }
        }
        if (IsLevelPassed(level2))
        {
            level2.SetActive(false);
            ChangeToNextLevel();
        }
    }

    else if (level3.getActive())
    {
        boardscore.text = level3.getScoreNeg().ToString();
        if (timer > 0)
        {
            timer -= Time.deltaTime; //Update timer
            if (timer <= 0)
            {
                level3.setPass(true);
            }
        }
    }
}

```

```

    }
    }
    if (IsLevelPassed(level3))
    {
        level3.setActive(false);
        board.SetActive(false);
        ChangeToNextLevel();
    }
}

else if (level4.getActive())
{
    boardscore.text = level4.getScoreNeg().ToString();
    if (timer > 0)
    {
        timer -= Time.deltaTime; //Update timer
        if (timer <= 0)
        {
            level4.setPass(true);
        }
    }
    if (IsLevelPassed(level4))
    {
        level4.setActive(false);
        ChangeToNextLevel();
    }
}

else if (level5.getActive())
{
    boardscore.text = level5.getScorePos().ToString();
    if (timer > 0)
    {
        timer -= Time.deltaTime; //Update timer
        if (timer <= 0)
        {
            level5.setPass(true);
        }
    }
    if (IsLevelPassed(level5))
    {
        level5.setActive(false);
        board.SetActive(false);
        ChangeToNextLevel();
    }
}
}

public bool IsLevelPassed(Level level)
{
    return level.getPass();
}

public void ChangeToNextLevel()
{
    if (SceneManager.sceneCount >= 3)
    {
        if (SceneManager.GetSceneAt(2).name == "Level1")
        {
            // Unload level1
            SceneManager.UnloadSceneAsync("Level1");
        }
    }
}

```



```

        setEvent(false);
        level2.SetActive(true);
        heartInventory.transform.Find("Item").gameObject.SetActive(true);
//Set the
        MainMenuManager.Instance.LevelEnd();
    }
    else if (SceneManager.GetSceneAt(2).name == "Level2")
    {
        SceneManager.UnloadSceneAsync("Level2");
        setEvent(false);
        level3.SetActive(true);
        MainMenuManager.Instance.LevelEnd();
    }
    else if (SceneManager.GetSceneAt(2).name == "Level3")
    {
        SceneManager.UnloadSceneAsync("Level3");
        setEvent(false);
        level4.SetActive(true);
        MainMenuManager.Instance.LevelEnd();
    }
    else if (SceneManager.GetSceneAt(2).name == "Level4")
    {
        SceneManager.UnloadSceneAsync("Level4");
        setEvent(false);
        level5.SetActive(true);
        MainMenuManager.Instance.LevelEnd();
    }
    else if (SceneManager.GetSceneAt(2).name == "Level5")
    {
        SceneManager.UnloadSceneAsync("Level5");
        setEvent(false);
        MainMenuManager.Instance.LevelEnd();
    }
}
}
}
public void setIsDialog(bool isdialog)
{
    this.isdialog = isdialog;
}

public bool getisDialog()
{
    return isdialog;
}

public void resetTimer(int num = 0)
{
    this.timer = duration + num;
}
public void openOption()
{
    optionCanvas.SetActive(true);
    Time.timeScale = 0f;
}

public void closeOption()
{
    optionCanvas.SetActive(false);
}

```

```

        Time.timeScale = 1f;
    }

    public void exitOption()
    {
        // Quits the application
        Application.Quit();
        // If you are in the editor, stop playing
#ifdef UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#endif
    }

    public void nextLevel(string level)
    {
        SceneManager.LoadSceneAsync(level, LoadSceneMode.Additive);
    }

    public void setEvent(bool flag)
    {
        eventSystem.SetActive(flag);
    }

    public void calScore()
    {
        this.currentScore = level1.getScorePos() - level1.getScoreNeg() +
        level2.getScorePos() - level2.getScoreNeg() + level3.getScorePos()*2 +
        level3.getScoreNeg() + level4.getScoreNeg() + level4.getScorePos() +
        level4.getItemScore() + level5.getScorePos();
    }

    public int getScore()
    {
        return currentScore;
    }
}

```

Menu Manager Script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using TMPro;

public class MainMenuManager : MonoBehaviour
{
    public static MainMenuManager Instance { get; private set; }
    [SerializeField] private GameObject mainMenuCanvas;
    [SerializeField] private GameObject Level1;
    [SerializeField] private GameObject Level2;
    [SerializeField] private GameObject Level3;
    [SerializeField] private GameObject Level4;
    [SerializeField] private GameObject Level5;

    private GameObject Level1IntroCanvas;
    private GameObject Level1KnowledgeCanvas;
    private GameObject Level1TutorialCanvas;
    private GameObject Level1ScoreCanvas;
    private GameObject Level1EndCanvas;

    private GameObject Level2IntroCanvas;
    private GameObject Level2KnowledgeCanvas;
    private GameObject Level2ScoreCanvas;
    private GameObject Level2EndCanvas;

    private GameObject Level3IntroCanvas;
    private GameObject Level3KnowledgeCanvas;
    private GameObject Level3ScoreCanvas;
    private GameObject Level3EndCanvas;

    private GameObject Level4IntroCanvas;
    private GameObject Level4KnowledgeCanvas;
    private GameObject Level4ScoreCanvas;
    private GameObject Level4EndCanvas;

    private GameObject Level5IntroCanvas;
    private GameObject Level5KnowledgeCanvas;
    private GameObject Level5ScoreCanvas;
    private GameObject Level5EndCanvas;
    private GameObject Level5Ranking;

    public void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
        LevelEnd();

        //Level 1 initialization
        Level1IntroCanvas = Level1.transform.Find("Level1-Intro").gameObject;
    }
}
```

```

        Level1KnowledgeCanvas = Level1.transform.Find("Level1-
Knowledge").gameObject;
        Level1TutorialCanvas = Level1.transform.Find("Level1-Tutorial").gameObject;
        Level1ScoreCanvas = Level1.transform.Find("Level1-Score").gameObject;
        Level1EndCanvas = Level1.transform.Find("Level1-End").gameObject;

        //Level 2 initialization
        Level2IntroCanvas = Level2.transform.Find("Level2-Intro").gameObject;
        Level2KnowledgeCanvas = Level2.transform.Find("Level2-
Knowledge").gameObject;
        Level2ScoreCanvas = Level2.transform.Find("Level2-Score").gameObject;
        Level2EndCanvas = Level2.transform.Find("Level2-End").gameObject;

        //Level 3 initialization
        Level3IntroCanvas = Level3.transform.Find("Level3-Intro").gameObject;
        Level3KnowledgeCanvas = Level3.transform.Find("Level3-
Knowledge").gameObject;
        Level3ScoreCanvas = Level3.transform.Find("Level3-Score").gameObject;
        Level3EndCanvas = Level3.transform.Find("Level3-End").gameObject;

        //Level 4 initialization
        Level4IntroCanvas = Level4.transform.Find("Level4-Intro").gameObject;
        Level4KnowledgeCanvas = Level4.transform.Find("Level4-
Knowledge").gameObject;
        Level4ScoreCanvas = Level4.transform.Find("Level4-Score").gameObject;
        Level4EndCanvas = Level4.transform.Find("Level4-End").gameObject;

        //Level 5 initialization
        Level5IntroCanvas = Level5.transform.Find("Level5-Intro").gameObject;
        Level5KnowledgeCanvas = Level5.transform.Find("Level5-
Knowledge").gameObject;
        Level5ScoreCanvas = Level5.transform.Find("Level5-Score").gameObject;
        Level5EndCanvas = Level5.transform.Find("Level5-End").gameObject;
        Level5Ranking = Level5.transform.Find("Level5-Rank").gameObject;
    }
    public void StartGame()
    {
        // Load the game scene by index or name (make sure it's added to build
settings)
        Level1TutorialCanvas.SetActive(true);
        mainMenuCanvas.SetActive(false);
    }

    public void QuitGame()
    {
        // Quits the application
        Application.Quit();
        // If you are in the editor, stop playing
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#endif
    }
    public void Level1Skip()
    {
        Level1IntroCanvas.SetActive(false);
        Level1KnowledgeCanvas.SetActive(true);
    }
    public void Level1SkipKnowledge()
    {
        Level1KnowledgeCanvas.SetActive(false);
        InvokeLevelManager.Instance.activeManager();
    }

```

```

}
public void Level1Start()
{
    Level1TutorialCanvas.SetActive(false);
    Level1IntroCanvas.SetActive(true);
}

public void Level1Next()
{
    Level1ScoreCanvas.SetActive(false);
    Level1EndCanvas.SetActive(true);

    Transform container = Level1EndCanvas.transform.Find("Background");
    Transform score = container.Find("score");
    if (score != null)
    {
        TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
        LevelManager.Instance.calScore();
        if (txt != null)
            txt.text = LevelManager.Instance.getScore().ToString();
    }
}

public void Level1End()
{
    Level1EndCanvas.SetActive(false);
    Level2IntroCanvas.SetActive(true);
}

public void Level2Knowledge()
{
    Level2IntroCanvas.SetActive(false);
    Level2KnowledgeCanvas.SetActive(true);
}

public void Level2Start()
{
    Level2KnowledgeCanvas.SetActive(false);
    LevelManager.Instance.setEvent(true);
    LevelManager.Instance.currentLevel = "Level2";
    LevelManager.Instance.level2.SetActive(true);
    SceneManager.LoadSceneAsync("Level2", LoadSceneMode.Additive);
    LevelManager.Instance.resetTimer();
}

public void Level2Next()
{
    Level2ScoreCanvas.SetActive(false);
    Level2EndCanvas.SetActive(true);

    Transform container = Level2EndCanvas.transform.Find("Background");
    Transform score = container.Find("score");
    if (score != null)
    {
        TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
        LevelManager.Instance.calScore();
        if (txt != null)
            txt.text = LevelManager.Instance.getScore().ToString();
    }
}

```

```

}

public void Level3Intro()
{
    Level2EndCanvas.SetActive(false);
    Level3IntroCanvas.SetActive(true);
}

public void Level3Knowledge()
{
    Level3IntroCanvas.SetActive(false);
    Level3KnowledgeCanvas.SetActive(true);
}

public void Level3Start()
{
    Level3KnowledgeCanvas.SetActive(false);
    LevelManager.Instance.setEvent(true);
    LevelManager.Instance.board.SetActive(true);
    LevelManager.Instance.currentLevel = "Level3";
    LevelManager.Instance.level3.SetActive(true);
    SceneManager.LoadSceneAsync("Level3", LoadSceneMode.Additive);
    LevelManager.Instance.resetTimer();
}

public void Level3End()
{
    Level3ScoreCanvas.SetActive(false);
    Level3EndCanvas.SetActive(true);
    Transform container = Level3EndCanvas.transform.Find("Background");
    Transform score = container.Find("score");
    if (score != null)
    {
        TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
        LevelManager.Instance.calScore();
        if (txt != null)
            txt.text = LevelManager.Instance.getScore().ToString();
    }
}

public void Level4Intro()
{
    Level3EndCanvas.SetActive(false);
    Level4IntroCanvas.SetActive(true);
}

public void Level4Knowledge()
{
    Level4IntroCanvas.SetActive(false);
    Level4KnowledgeCanvas.SetActive(true);
}

public void Level4Start()
{
    Level4KnowledgeCanvas.SetActive(false);
    LevelManager.Instance.setEvent(true);
    LevelManager.Instance.currentLevel = "Level4";
    LevelManager.Instance.level4.SetActive(true);
    SceneManager.LoadSceneAsync("Level4", LoadSceneMode.Additive);
    LevelManager.Instance.resetTimer();
}

```

```

public void Level4End()
{
    Level4ScoreCanvas.SetActive(false);
    Level4EndCanvas.SetActive(true);
    Transform container = Level4EndCanvas.transform.Find("Background");
    Transform score = container.Find("score");
    if (score != null)
    {
        TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
        LevelManager.Instance.calScore();
        if (txt != null)
            txt.text = LevelManager.Instance.getScore().ToString();
    }
}

public void Level5Intro()
{
    Level4EndCanvas.SetActive(false);
    Level5IntroCanvas.SetActive(true);
}

public void Level5Knowledge()
{
    Level5IntroCanvas.SetActive(false);
    Level5KnowledgeCanvas.SetActive(true);
}

public void Level5Start()
{
    Level5KnowledgeCanvas.SetActive(false);
    LevelManager.Instance.setEvent(true);
    LevelManager.Instance.board.SetActive(true);
    LevelManager.Instance.level5.SetActive(true);
    PlayerController.Instance.resetPos();
    SceneManager.LoadSceneAsync("Level5", LoadSceneMode.Additive);
    LevelManager.Instance.resetTimer(200);
    LevelManager.Instance.currentLevel = "Level5";
}

public void Level5End()
{
    Level5ScoreCanvas.SetActive(false);
    Level5EndCanvas.SetActive(true);
    Transform container = Level5EndCanvas.transform.Find("Background");
    Transform score = container.Find("score");
    if (score != null)
    {
        TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
        LevelManager.Instance.calScore();
        if (txt != null)
            txt.text = LevelManager.Instance.getScore().ToString();
    }
}

public void ToRanking()
{
    Level5EndCanvas.SetActive(false);
    Level5Ranking.SetActive(true);
}

```

```

int myScore = LevelManager.Instance.getScore();
Transform container = Level5Ranking.transform.Find("Background");
Transform score = container.Find("score");
Transform description = container.Find("Description");
if (myScore >= 100)
{
    TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Perfect";
    txt = description.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Outstanding performance! You fully understand the
effects of obesity on body organs and their functions, demonstrating comprehensive
knowledge throughout the game.";
}
else if (myScore >= 80)
{
    TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Great";
    txt = description.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Impressive! You have a strong grasp of the impact of
obesity on the body's organs and functions, missing only a few minor details.";
}
else if (myScore >= 60)
{
    TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Well";
    txt = description.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Good effort! You show a reasonable understanding of the
effects of obesity on organ functions, but there are still areas to improve.";
}
else if (myScore >= 40)
{
    TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Satisfactory";
    txt = description.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Decent work. You have a basic understanding of how
obesity affects the body, but there are several aspects that require further
attention.";
}
else
{
    TextMeshProUGUI txt = score.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "Unsatisfactory";
    txt = description.GetComponent<TextMeshProUGUI>();
    if (txt != null)
        txt.text = "There's room for improvement. Your understanding of the
effects of obesity on body organs and functions is limited, and further study is
recommended.";
}
}
public void ToEnding()
{

```



```

        Level5Ranking.SetActive(false);
        SceneManager.LoadSceneAsync("Ending", LoadSceneMode.Additive);
    }

    public void LevelEnd()
    {
        if (LevelManager.Instance != null)
        {
            if (LevelManager.Instance.currentLevel == "Level1")
            {
                Level1ScoreCanvas.SetActive(true);
                Transform container =
Level1ScoreCanvas.transform.Find("Background");
                if (container != null)
                {
                    Transform noRBC = container.Find("noRBC");
                    Transform noEnemy = container.Find("noEnemy");
                    if (noRBC != null)
                    {
                        TextMeshProUGUI txt = noRBC.GetComponent<TextMeshProUGUI>();
                        if (txt != null)
                            txt.text =
LevelManager.Instance.level1.getScoreNeg().ToString();
                    }

                    if (noEnemy != null)
                    {
                        TextMeshProUGUI txt =
noEnemy.GetComponent<TextMeshProUGUI>();
                        if (txt != null)
                            txt.text =
LevelManager.Instance.level1.getScorePos().ToString();
                    }
                }
            }
            else if (LevelManager.Instance.currentLevel == "Level2")
            {
                Level2ScoreCanvas.SetActive(true);
                Transform container =
Level2ScoreCanvas.transform.Find("Background");
                if (container != null)
                {
                    Transform noVictim = container.Find("noVictim");
                    Transform noEnemy = container.Find("noEnemy");
                    if (noVictim != null)
                    {
                        TextMeshProUGUI txt =
noVictim.GetComponent<TextMeshProUGUI>();
                        if (txt != null)
                            txt.text =
LevelManager.Instance.level2.getScorePos().ToString();
                    }

                    if (noEnemy != null)
                    {
                        TextMeshProUGUI txt =
noEnemy.GetComponent<TextMeshProUGUI>();
                        if (txt != null)
                            txt.text =
LevelManager.Instance.level2.getScoreNeg().ToString();
                    }
                }
            }
        }
    }
}

```



```

        }

        if (novictim != null)
        {
            TextMeshProUGUI txt =
novictim.GetComponent<TextMeshProUGUI>();
            if (txt != null)
                txt.text =
LevelManager.Instance.level4.getScoreNeg().ToString();

            if (LevelManager.Instance.level4.getScoreNeg() > 0)
                txt.color = Color.green;
            else if (LevelManager.Instance.level4.getScoreNeg() < 0)
                txt.color = Color.red;
        }

        if (noitems != null)
        {
            TextMeshProUGUI txt =
noitems.GetComponent<TextMeshProUGUI>();
            if (txt != null)
                txt.text =
LevelManager.Instance.level4.getItemScore().ToString();

            if (LevelManager.Instance.level4.getItemScore() > 0)
                txt.color = Color.green;
            else if (LevelManager.Instance.level4.getItemScore() < 0)
                txt.color = Color.red;
        }
    }
}
else if (LevelManager.Instance.currentLevel == "Level5")
{
    Level5ScoreCanvas.SetActive(true);
    Transform container =
Level5ScoreCanvas.transform.Find("Background");
    if (container != null)
    {
        Transform number = container.Find("number");
        if (number != null)
        {
            TextMeshProUGUI txt =
number.GetComponent<TextMeshProUGUI>();
            if (txt != null)
                txt.text =
(LevelManager.Instance.level5.getScorePos()).ToString();
        }
    }
}
else
{
    mainMenuCanvas.SetActive(true);
}
}
}

```

Player Controller Script:

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

public class PlayerController : MonoBehaviour
{
    public bool FacingLeft { get { return facingLeft; } set { facingLeft = value; } }
}

public bool FacingUp { get { return facingUp; } set { facingUp = value; } }
public static PlayerController Instance { get; private set; }

[SerializeField] private float moveSpeed = 0.1f;
[SerializeField] private float dashSpeed = 4f;
[SerializeField] private TrailRenderer myTrailRenderer;
[SerializeField] private Transform weaponCollider;
[SerializeField] private LayerMask collisionLayer; // Layer for collision
detection

private PlayerControls playerControls;
private Vector2 movement;
private Rigidbody2D rb;
private Animator myAnimator;
private SpriteRenderer mySpriteRender;
private bool canMove = true;

private bool facingLeft = false;
private bool facingUp = false;
private GameObject targetObject;
private void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(gameObject);
    }

    playerControls = new PlayerControls();
    rb = GetComponent<Rigidbody2D>();
    myAnimator = GetComponent<Animator>();
    mySpriteRender = GetComponent<SpriteRenderer>();

    // Subscribe to the move action
    playerControls.Player.Move.performed += ctx => movement =
ctx.ReadValue<Vector2>();
    playerControls.Player.Move.canceled += ctx => movement = Vector2.zero;
}

private void OnEnable()
{
    playerControls.Player.Enable();
}

private void Update()
```

```

{
    PlayerInput();
    Move();
}

private void FixedUpdate()
{
    AdjustPlayerFacingDirection();
}

public Transform GetWeaponCollider()
{
    return weaponCollider;
}

private void PlayerInput()
{
    // No need to read input here since it's already stored in the 'movement'
variable
    myAnimator.SetFloat("moveX", movement.x);
    myAnimator.SetFloat("moveY", movement.y);
}

private void Move()
{
    Vector2 newPosition = rb.position + movement * (moveSpeed *
Time.fixedDeltaTime);
    if (canMove && CanMoveToPosition(newPosition))
    {
        rb.MovePosition(newPosition);
    }
}

private void AdjustPlayerFacingDirection()
{
    float movementX = movement.x;
    float movementY = movement.y;
    Vector3 playerScreenPoint =
Camera.main.WorldToScreenPoint(transform.position);

    if (movementX < 0)
    {
        mySpriteRender.flipX = true;
        FacingLeft = true;
        myAnimator.SetBool("ismoveX", true);
    }
    else if (movementX > 0)
    {
        mySpriteRender.flipX = false;
        FacingLeft = false;
        myAnimator.SetBool("ismoveX", true);
    }
    else
    {
        myAnimator.SetBool("ismoveX", false);
    }

    if (movementY > 0)
    {
        FacingUp = true;
        myAnimator.SetBool("ismoveYp", true);
    }
}

```

```

    }
    else if (movementY < 0)
    {
        mySpriteRender.flipY = false;
        FacingUp = false;
        myAnimator.SetBool("ismoveY", true);
    }
    else
    {
        myAnimator.SetBool("ismoveY", false);
        myAnimator.SetBool("ismoveYp", false);
    }
}

public PlayerControls getPlayerControls()
{
    return playerControls;
}

private bool CanMoveToPosition(Vector2 newPosition)
{
    Vector2 direction = (newPosition - (Vector2)transform.position).normalized;
    float distance = Vector2.Distance(transform.position, newPosition);

    RaycastHit2D hit = Physics2D.Raycast(transform.position, direction,
distance, collisionLayer);

    if (hit.collider != null)
    {
        return false;
    }

    return true;
}

public void SetMove(bool flag)
{
    this.canMove = flag;
}

public void resetPos()
{
    transform.position = Vector3.zero;
}

//Pickups & Drop
private void OnTriggerEnter2D(Collider2D other)
{
    if (other.CompareTag("Target"))
    {
        targetObject = other.gameObject;
        LevelManager.Instance.inventory.SetActive(false);
        LevelManager.Instance.pickupbtn.SetActive(true);
    }
    else if (other.CompareTag("NotTarget"))
    {
        targetObject = other.gameObject;
        LevelManager.Instance.inventory.SetActive(false);
        LevelManager.Instance.pickupbtn.SetActive(true);
    }
    else if (other.CompareTag("box"))

```

```

    {
        Animator anime = other.GetComponent<Animator>();
        if (anime != null)
        {
            anime.SetTrigger("open");
        }
        targetObject = other.gameObject;
        LevelManager.Instance.inventory.SetActive(false);
        LevelManager.Instance.droptbn.SetActive(true);
    }
}

private void OnTriggerExit2D(Collider2D other)
{
    if (other.CompareTag("Target"))
    {
        LevelManager.Instance.inventory.SetActive(true);
        LevelManager.Instance.pickupbtn.SetActive(false);
        targetObject = null;
    }
    else if (other.CompareTag("NotTarget"))
    {
        LevelManager.Instance.inventory.SetActive(true);
        LevelManager.Instance.pickupbtn.SetActive(false);
        targetObject = null;
    }
    else if (other.CompareTag("box"))
    {
        LevelManager.Instance.inventory.SetActive(true);
        LevelManager.Instance.droptbn.SetActive(false);
        targetObject = null;
    }
}

public void OnPickupButton()
{
    if (targetObject != null)
    {
        if (LevelManager.Instance.currentLevel == "Level3")
        {
            LevelManager.Instance.level3.addScoreNeg();
        }
        else if (LevelManager.Instance.currentLevel == "Level4")
        {
            if (targetObject.CompareTag("Target"))
                LevelManager.Instance.level4.addItemScore();
            else if (targetObject.CompareTag("NotTarget"))
                LevelManager.Instance.level4.deductItemScore();
        }
        else if (LevelManager.Instance.currentLevel == "Level5")
        {
            LevelManager.Instance.level5.addScorePos();
        }
        Destroy(targetObject);
    }
}

public void OnDropButton()
{
    if (targetObject != null)
    {
        if (LevelManager.Instance.currentLevel == "Level3")
        {

```

```

LevelManager.Instance.level3.setScorePos(LevelManager.Instance.level3.getScorePos()
+ LevelManager.Instance.level3.getScoreNeg());
        LevelManager.Instance.level3.setScoreNeg(0);
    }
}
}
}

```

Active Weapon Script:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ActiveWeapon : MonoBehaviour
{
    public static ActiveWeapon Instance { get; private set; }

    [SerializeField] private MonoBehaviour currentActiveWeapon;

    private PlayerControls playerControls;
    public Animator myAnimator;
    private bool isAttacking = false;
    private bool attackButtonDown = false;

    private void Awake()
    {
        // Singleton pattern implementation
        if (Instance != null && Instance != this)
        {
            Destroy(gameObject);
        }
        else
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }

        playerControls = new PlayerControls();
        myAnimator = GetComponent<Animator>();
    }

    private void OnEnable()
    {
        playerControls.Enable();
    }

    private void Start()
    {
        playerControls.Combat.Attack.started += _ => StartAttacking();
        playerControls.Combat.Attack.canceled += _ => StopAttacking();
    }

    private void Update()
    {
        Attack();
    }
}

```



```

public PlayerControls getPlayerControls()
{
    return playerControls;
}
public void NewWeapon(MonoBehaviour newWeapon)
{
    currentActiveWeapon = newWeapon;
}

public void WeaponNull()
{
    currentActiveWeapon = null;
}

public MonoBehaviour getWeapon()
{
    return currentActiveWeapon;
}

public void ToggleIsAttacking(bool value)
{
    isAttacking = value;
}

private void StartAttacking()
{
    attackButtonDown = true;
}

private void StopAttacking()
{
    attackButtonDown = false;
}

private void Attack()
{
    if (attackButtonDown && !isAttacking)
    {
        isAttacking = true;
    }
}
}

```

Sword Script:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Sword : MonoBehaviour
{
    [SerializeField] private GameObject slashAnimPrefab;
    [SerializeField] private Transform slashAnimSpawnPoint;

    // Start is called before the first frame update
    private Transform weaponCollider;
    private PlayerControls playerControls;
    private Animator myAnimator;
    private bool isFiring = false; // Firing flag
}

```

```

private GameObject slashAnim;
private void Awake()
{
    myAnimator = GetComponent<Animator>();
    playerControls = new PlayerControls();
    slashAnimSpawnPoint = this.transform; // Set the spawn point to the Apple's
transform
}

private void OnEnable()
{
    playerControls.Enable();
}

private void OnDisable()
{
    playerControls.Disable();
}

private void Start()
{
    playerControls.Combat.Attack.started += _ => Attack();
    weaponCollider = PlayerController.Instance.GetWeaponCollider();
}

private void Update()
{
    MouseFollowWithOffset();
}

private void Attack()
{
    isFiring = true; // Set the firing flag
    if (myAnimator != null)
    {
        myAnimator.SetTrigger("Attack");
    }
    else
    {
        Awake();
    }
    slashAnim = Instantiate(slashAnimPrefab, slashAnimSpawnPoint.position,
Quaternion.identity);
    slashAnim.transform.parent = this.transform.parent;
}

public void EndAttack()
{
    isFiring = false; // Reset the firing flag after the attack animation
}

public void SwingUpFlipAnim()
{
    this.slashAnim.gameObject.transform.rotation = Quaternion.Euler(-100, 0, 0);

    if (PlayerController.Instance.FacingLeft)
    {
        slashAnim.GetComponent<SpriteRenderer>().flipX = true;
    }
}

```

```

public void SwingDownFlipAnim()
{
    this.slashAnim.gameObject.transform.rotation = Quaternion.Euler(-100, 0, 0);

    if (!PlayerController.Instance.FacingLeft)
    {
        this.slashAnim.GetComponent<SpriteRenderer>().flipX = false;
    }
}

private void MouseFollowWithOffset()
{
    Vector3 mousePos = Input.mousePosition;
    Vector3 playerScreenPoint =
Camera.main.WorldToScreenPoint(PlayerController.Instance.transform.position);

    float angle = Mathf.Atan2(mousePos.y, mousePos.x) * Mathf.Rad2Deg;

    if (mousePos.x < playerScreenPoint.x)
    {
        ActiveWeapon.Instance.transform.rotation = Quaternion.Euler(0, -180,
angle);
    }
    else
    {
        ActiveWeapon.Instance.transform.rotation = Quaternion.Euler(0, 0,
angle);
    }
}
}

```

Heart Script:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Heart : MonoBehaviour
{
    [SerializeField] private GameObject slashAnimPrefab;
    [SerializeField] private Transform slashAnimSpawnPoint;

    // Start is called before the first frame update
    private Transform weaponCollider;
    private PlayerControls playerControls;
    private Animator myAnimator;
    private bool isFiring = false; // Firing flag
    private GameObject slashAnim;
    private void Awake()
    {
        myAnimator = GetComponent<Animator>();
        playerControls = new PlayerControls();
        slashAnimSpawnPoint = this.transform; // Set the spawn point to the Apple's
transform
    }
}

```

```

private void OnEnable()
{
    playerControls.Enable();
}

private void OnDisable()
{
    playerControls.Disable();
}

private void Start()
{
    playerControls.Combat.Attack.started += _ => Attack();
}

private void Update()
{
    MouseFollowWithOffset();
}

private void Attack()
{
    isFiring = true; // Set the firing flag
    if (myAnimator != null)
    {
        myAnimator.SetTrigger("Attack");
    }
    else
    {
        Awake();
    }
    slashAnim = Instantiate(slashAnimPrefab, slashAnimSpawnPoint.position,
Quaternion.identity);
    slashAnim.transform.parent = this.transform.parent;
}

public void SwingUpFlipAnim()
{
    slashAnim.gameObject.transform.rotation = Quaternion.Euler(-100, 0, 0);

    if (PlayerController.Instance.FacingLeft)
    {
        slashAnim.GetComponent<SpriteRenderer>().flipX = true;
    }
}

public void SwingDownFlipAnim()
{
    slashAnim.gameObject.transform.rotation = Quaternion.Euler(-100, 0, 0);

    if (!PlayerController.Instance.FacingLeft)
    {
        slashAnim.GetComponent<SpriteRenderer>().flipX = false;
    }
}

private void MouseFollowWithOffset()
{
    Vector3 mousePos = Input.mousePosition;
}

```

```

    Vector3 playerScreenPoint =
Camera.main.WorldToScreenPoint(PlayerController.Instance.transform.position);

    float angle = Mathf.Atan2(mousePos.y, mousePos.x) * Mathf.Rad2Deg;

    if (mousePos.x < playerScreenPoint.x)
    {
        ActiveWeapon.Instance.transform.rotation = Quaternion.Euler(0, -180,
angle);
    }
    else
    {
        ActiveWeapon.Instance.transform.rotation = Quaternion.Euler(0, 0,
angle);
    }
}
}

```

Enemy Spawn Script:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class EnemySpawn1 : MonoBehaviour
{
    [SerializeField] private GameObject enemyPrefab;
    [SerializeField] private float minSpawnTime;
    [SerializeField] private float maxSpawnTime;
    [SerializeField] private string levelname;
    private float timeToSpawn;
    void Awoke()
    {
        setTimeToSpawn();
    }

    // Update is called once per frame
    void Update()
    {
        timeToSpawn -= Time.deltaTime;
        if (timeToSpawn <= 0)
        {
            Scene targetScene = SceneManager.GetSceneByName(levelname);
            if (!targetScene.IsValid())
            {
                Debug.LogError("Target scene is not loaded.");
                return;
            }

            GameObject enemyInstance = Instantiate(enemyPrefab, transform.position,
Quaternion.identity);
            // Move the instantiated object to the target scene
            SceneManager.MoveGameObjectToScene(enemyInstance, targetScene);
            setTimeToSpawn();
        }
    }

    private void setTimeToSpawn()

```

```
{
    timeToSpawn = Random.Range(minSpawnTime, maxSpawnTime);
}
}
```

Level Script:

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

public class Level
{
    private int scorePos = 0;
    private int scoreNeg = 0;
    private bool pass = false;
    private bool active = false;
    private int scoreitem = 0;

    public Level()
    {}
    public Level(int scorePos,int scoreNeg,bool pass,bool active)
    {
        this.scorePos = scorePos;
        this.scoreNeg = scoreNeg;
        this.pass = pass;
        this.active = active;
    }

    public int getScorePos()
    {
        return scorePos;
    }

    public int getScoreNeg()
    {
        return scoreNeg;
    }

    public void setScorePos(int scorePos)
    {
        this.scorePos = scorePos;
    }
    public void setScoreNeg(int scoreNeg)
    {
        this.scoreNeg = scoreNeg;
    }

    public void addScorePos()
    {
        this.scorePos += 1;
    }
    public void addScoreNeg()
    {
        this.scoreNeg += 1;
    }

    public void deductScoreNeg()
```

```

    {
        this.scoreNeg -= 1;
    }

    public void deductScorePos()
    {
        this.scorePos -= 1;
    }

    public bool getPass()
    {
        return pass;
    }

    public void setPass(bool pass)
    {
        this.pass = pass;
    }

    public bool getActive()
    {
        return active;
    }

    public void setActive(bool active)
    {
        this.active = active;
    }

    public void addItemScore()
    {
        this.scoreitem += 1;
    }

    public void deductItemScore()
    {
        this.scoreitem -= 1;
    }

    public void setItemScore(int score)
    {
        this.scoreitem = score;
    }

    public int getItemScore()
    {
        return this.scoreitem;
    }
}

```

Damage Sources Script:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DamageSource : MonoBehaviour
{
    [SerializeField] private int damageAmount = 1;
}

```

```

private void OnTriggerEnter2D(Collider2D other)
{
    GameObject weapon = this.gameObject;
    Sword isSword = weapon.GetComponentInChildren<Sword>();
    Heart isHeart = weapon.GetComponentInChildren<Heart>();
    if (isSword != null)
    {
        // If the game object has the EnemyAI component
        EnemyHealth enemyhealth = other.gameObject.GetComponent<EnemyHealth>();
        VictimHealth victimhealth =
other.gameObject.GetComponent<VictimHealth>();
        Victim victim = other.gameObject.GetComponent<Victim>();

        if (enemyhealth != null)
        {
            enemyhealth.TakeDamage(damageAmount);
            if (LevelManager.Instance.currentLevel == "Level1")
            {
                LevelManager.Instance.level1.addScorePos();
            }
            else if (LevelManager.Instance.currentLevel == "Level2")
            {
                LevelManager.Instance.level2.addScorePos();
            }
            else if (LevelManager.Instance.currentLevel == "Level4")
            {
                LevelManager.Instance.level4.addScorePos();
            }
        }

        if (victim != null)
        {
            Animator anime = victim.GetComponent<Animator>();

            if (anime != null)
            {
                anime.SetTrigger("hurt");
                if (LevelManager.Instance.currentLevel == "Level1")
                {
                    LevelManager.Instance.level1.addScoreNeg();
                }
                else if (LevelManager.Instance.currentLevel == "Level2")
                {
                    anime.SetInteger("health", anime.GetInteger("health") - 1);
                    LevelManager.Instance.level2.addScoreNeg();
                }
                else if (LevelManager.Instance.currentLevel == "Level4")
                {
                    anime.SetInteger("health", anime.GetInteger("health") - 1);
                    LevelManager.Instance.level4.deductScoreNeg();
                }
            }

            if (victimhealth != null)
            {
                victimhealth.TakeDamage(damageAmount); //cause damage
            }
        }
    }
}

```



```

    }
    else if (isHeart != null)
    {
        EnemyHealth enemyhealth = other.gameObject.GetComponent<EnemyHealth>();
        VictimHealth victimhealth =
other.gameObject.GetComponent<VictimHealth>();

        if (victimhealth != null)
        {
            victimhealth.Takeheal(damageAmount);
            Victim victim = other.gameObject.GetComponent<Victim>();
            Animator anime = victim.GetComponent<Animator>();
            if (anime != null)
            {
                //set anime
                if (LevelManager.Instance.currentLevel == "Level1")
                {
                    LevelManager.Instance.level1.addScorePos();
                }
                else if (LevelManager.Instance.currentLevel == "Level2")
                {
                    anime.SetInteger("health", anime.GetInteger("health") + 1);
                    LevelManager.Instance.level2.addScorePos();
                }
                else if (LevelManager.Instance.currentLevel == "Level4")
                {
                    anime.SetInteger("health", anime.GetInteger("health") + 1);
                    LevelManager.Instance.level4.addScoreNeg();
                }
            }
        }
        if (enemyhealth != null)
        {
            enemyhealth.Takeheal(damageAmount);
            Transform enemy = other.gameObject.GetComponent<Transform>();
            if (LevelManager.Instance.currentLevel == "Level1")
            {
                LevelManager.Instance.level1.addScoreNeg();
            }
            else if (LevelManager.Instance.currentLevel == "Level2")
            {
                LevelManager.Instance.level2.addScoreNeg();
                enemy.transform.localScale *= 1.2f; //scale factor
            }
            else if (LevelManager.Instance.currentLevel == "Level4")
            {
                LevelManager.Instance.level4.deductScorePos();
                enemy.transform.localScale *= 1.2f; //scale factor
            }
        }
    }
    else
    {
        Debug.Log("Null weapon!");
        Debug.Log("This script is attached to: " + gameObject.name);
    }
}
}

```

Enemy AI Script:

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

public class EnemyAI : MonoBehaviour
{
    private enum State
    {
        Roaming
    }

    private State state;
    private EnemyFindPath enemyfindpath;
    private void Awake()
    {
        enemyfindpath = GetComponent<EnemyFindPath>();
        state = State.Roaming;
    }

    private void Start()
    {
        StartCoroutine(RoamingRoutine());
    }

    private IEnumerator RoamingRoutine()
    {
        while(state == State.Roaming)
        {
            Vector2 roamPosition = GetRoamingPosition();
            enemyfindpath.MoveTo(roamPosition);
            yield return new WaitForSeconds(2f);
        }
    }

    private Vector2 GetRoamingPosition()
    {
        return new Vector2(Random.Range(-1f, 1f), Random.Range(-1f, 1f)).normalized;
    }
}
```

Enemy Health Script:

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

public class EnemyHealth : MonoBehaviour
{
    [SerializeField] private int startingHealth = 3;

    private int currentHealth;
    private KnockBack knockback;
    LevelManager levelManager;

    private void Awake()
    {
        knockback = GetComponent<KnockBack>();
    }
}
```

```

        levelManager = LevelManager.Instance;
    }

    private void Start()
    {
        currentHealth = startingHealth;
    }

    public void TakeDamage(int damage)
    {
        if (!levelManager.getisDialog())
        {
            currentHealth -= damage;
            //Debug.Log(PlayerController.Instance.transform.position);
            knockback.GetKnockedBack(PlayerController.Instance.transform, 100.0f);
            DetectDeath();
        }
    }

    public void Takeheal(int heal)
    {
        if (!levelManager.getisDialog())
        {
            currentHealth += heal;
            DetectDeath();
        }
    }

    private void DetectDeath()
    {
        if (currentHealth <= 0)
        {
            Destroy(gameObject);
        }
    }
}

```

Victim Health Script:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class VictimHealth : MonoBehaviour
{
    [SerializeField] private int startingHealth = 3;
    private int currentHealth;

    private void Start()
    {
        currentHealth = startingHealth;
    }

    public void TakeDamage(int damage)
    {
        if (!LevelManager.Instance.getisDialog())
        {
            currentHealth -= damage;
        }
    }
}

```

```

        DetectDeath();
    }
}

public void Takeheal(int heal)
{
    if (!LevelManager.Instance.getisDialog())
    {
        currentHealth += heal;
        DetectDeath();
    }
}

private void DetectDeath()
{
    if (currentHealth <= 0) //victim health drop to zero
    {
        Destroy(gameObject);
    }
}
}

```

Dialogs Script:

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using TMPro;

public class Dialogs : MonoBehaviour
{
    public TextMeshProUGUI textComponent;
    public float textSpeed;
    private string[] lines;
    private int index;

    // Set the dialog file for the Dialogs script
    public void SetDialogFile(TextAsset file)
    {
        dialogFile = file;
        LoadDialogFromFile();
    }

    private TextAsset dialogFile; // txt file

    public void Start()
    {
        textComponent.text = string.Empty;
        StartDialog();
        LevelManager.Instance.setIsDialog(true);
        PlayerController.Instance.SetMove(false);
    }

    public void Update()
    {
        if(LevelManager.Instance.getisDialog()) //if is dialogue
            LevelManager.Instance.resetTimer();
        if (Input.GetMouseButtonDown(0))

```

```

        {
            if (textComponent.text == lines[index])
            {
                NextLine();
            }
            else
            {
                StopAllCoroutines();
                textComponent.text = lines[index];
            }
        }
    }

void LoadDialogFromFile()
{
    if (dialogFile != null)
    {
        // Split the text into lines based on newline character '\n'
        lines = dialogFile.text.Split('\n');
    }
    else
    {
        Debug.LogError("Dialog file is not assigned.");
    }
}

void StartDialog()
{
    index = 0;
    StartCoroutine(TypeLine());
}

IEnumerator TypeLine()
{
    if (index >= lines.Length)
    {
        yield break; // Exit the coroutine if index is out of bounds
    }

    foreach (char c in lines[index].ToCharArray())
    {
        textComponent.text += c;
        yield return new WaitForSeconds(textSpeed);
    }
}

public void NextLine()
{
    if (index < lines.Length - 1)
    {
        index++;
        textComponent.text = string.Empty;
        StartCoroutine(TypeLine());
    }
    else
    {
        gameObject.SetActive(false);
        LevelManager.Instance.setIsDialog(false);
        PlayerController.Instance.SetMove(true);
        if (LevelManager.Instance.currentLevel == "Level5")
            LevelManager.Instance.resetTimer(60); //The timer for level 5
    }
}

```

```
}  
  }  
}
```

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y3	Study week no.: 3
Student Name & ID: Pang Qing Sen & 2106605	
Supervisor: Ts Dr Tong Dong Ling	
Project Title: Gamification in Learning Obesity	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- First level of the game and weapon behavior
- Player behavior and enemy behavior
- Some animation of the player and enemy

2. WORK TO BE DONE

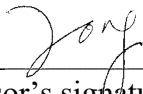
- UI of the weapon system
- Design and second level of the game
- Weapon changing and behavior with other game object

3. PROBLEMS ENCOUNTERED

- Some of the bugs faced when need to spawn new weapon
- Finding of the resources of the rest of the levels
- Collision detection system is not very sensitive

4. SELF EVALUATION OF THE PROGRESS

- The progress considered slow, might need to chase the progress to develop the whole system.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y3	Study week no.: 5
Student Name & ID: Pang Qing Sen & 2106605	
Supervisor: Ts Dr Tong Dong Ling	
Project Title: Gamification in Learning Obesity	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Second level of the system
- UI of the weapon system

2. WORK TO BE DONE

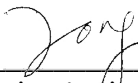
- Third level of the system
- Weapon behavior with other game object

3. PROBLEMS ENCOUNTERED


- Error occurs when spawn new weapon
- Error when loading new levels

4. SELF EVALUATION OF THE PROGRESS

- Still on schedule, but might require further study on the bugs faced



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y3	Study week no.: 7
Student Name & ID: Pang Qing Sen & 2106605	
Supervisor: Ts Dr Tong Dong Ling	
Project Title: Gamification in Learning Obesity	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Weapon behavior with other game object
- Level 3, Level 4

2. WORK TO BE DONE

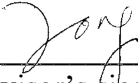
- Pickup system of the game
- UI for the game
- Score calculation system

3. PROBLEMS ENCOUNTERED


- Level 3 and 4 require the pickup system to work
- Bug was fixed but now with some warning
- Some of the image is blur when putting inside unity

4. SELF EVALUATION OF THE PROGRESS

- The progress is on schedule, but need to chase to finish the Pickup system since it is important for both Level 3 and 4



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y3	Study week no.: 8
Student Name & ID: Pang Qing Sen & 2106605	
Supervisor: Ts Dr Tong Dong Ling	
Project Title: Gamification in Learning Obesity	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Pickup system of the game
- Score calculation system
- UI of the game includes the menu and pages

2. WORK TO BE DONE

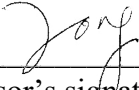
- Level 5 & Ending
- Timer for the game

3. PROBLEMS ENCOUNTERED


- Finding the resources of the UI
- The changing of the pages which the event listener located in different scene

4. SELF EVALUATION OF THE PROGRESS

- The progress is slight behind schedule, might need to chase for the works.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: S3Y3	Study week no.: 11
Student Name & ID: Pang Qing Sen & 2106605	
Supervisor: Ts Dr Tong Dong Ling	
Project Title: Gamification in Learning Obesity	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Level 5 & Ending
- Timer for the game

2. WORK TO BE DONE


- Report writing
- The score ranking of the game
- Testing and validation of the game

3. PROBLEMS ENCOUNTERED

- Report writing
- The score ranking of the game

4. SELF EVALUATION OF THE PROGRESS

- The progress still in schedule but need to work for the report writing and some of the validation and testing process of the game application.



Supervisor's signature



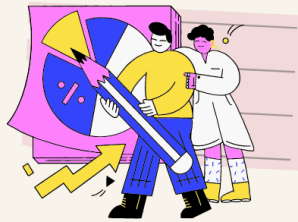
Student's signature

Gamification of Learning Obesity



Do You Ever Think Game Can Help You To Learn?

Via this game project, you will gain a general knowledge of the effect of obesity on our organs through different levels of the game.



Our Objectives & Scopes

To let people have a better understanding of the effect of obesity on our body organs via the developed 2D game using Unity Engine.



Our Methodologies

Implemented various game elements such as the levels design, dialogue system, scoring and reward system.



Conclusion

Try our developed game if you wish to have a better understanding of the effects of obesity. You will gain knowledge about it via this gameplay.

Bachelor of Computer Science (Hons)

By: Pang Qing Sen



Final Year Project

PLAGIARISM CHECK RESULT

Document Viewer

Turnitin Originality Report

Processed on: 11-Sep-2024 21:50 +08
ID: 2451012629
Word Count: 14774
Submitted: 1

Gamification in Learning Obesity By Pang Qing Sen

Similarity Index	Similarity by Source
3%	Internet Sources: 2% Publications: 1% Student Papers: 2%

include quoted	include bibliography	excluding matches < 8 words	mode: quickview (classic) report	print	download
<1% match (student papers from 10-Sep-2023) Class: UCCC2513 Assignment: T10_final report Paper ID: 2161885826					
<1% match (student papers from 14-Sep-2023) Submitted to Universiti Tunku Abdul Rahman on 2023-09-14					
<1% match (student papers from 25-Apr-2024) Submitted to Universiti Tunku Abdul Rahman on 2024-04-25					
<1% match (student papers from 16-Dec-2021) Submitted to University of Glasgow on 2021-12-16					
<1% match (publications) Franz Lanzinger, "3D Game Development with Unity", CRC Press, 2022					
<1% match (Sean Ong, Varun Kumar Siddaraju, "Beginning Windows Mixed Reality Programming", Springer Science and Business Media LLC, 2021) Sean Ong, Varun Kumar Siddaraju, "Beginning Windows Mixed Reality Programming", Springer Science and Business Media LLC, 2021					
<1% match (Internet from 23-Jun-2024) https://wolfcartel.it.ch/the-witchfinder					
<1% match (student papers from 27-Nov-2022) Submitted to University of Greenwich on 2022-11-27					
<1% match (student papers from 22-May-2024) Submitted to University of Central England in Birmingham on 2024-05-22					
<1% match (Internet from 18-Dec-2023) https://5dok.net/document/rz38648q-play-play-motivational-effects-games-engineering-education.html					
<1% match (student papers from 03-Jun-2024) Submitted to Macquarie University on 2024-06-03					
<1% match (student papers from 01-Jun-2023) Submitted to Nottingham Trent University on 2023-06-01					
<1% match (Internet from 05-Aug-2014) http://www.researchgate.net					
<1% match (student papers from 20-Apr-2020) Submitted to Seminole County Public Schools on 2020-04-20					
<1% match (Internet from 29-Apr-2019) https://www.genf20plus.info/metabolism.php					
<1% match (Dongliang Xu, Zaijun Wu, Junjun Xu, Qinran Hu. "A pseudo measurement modeling based forecasting aided state estimation framework for distribution network", International Journal of Electrical Power & Energy Systems, 2024) Dongliang Xu, Zaijun Wu, Junjun Xu, Qinran Hu. "A pseudo measurement modeling based forecasting aided state estimation framework for distribution network", International Journal of Electrical Power & Energy Systems, 2024					
<1% match (Internet from 13-Jan-2023) https://fse.studenttheses.ub.rug.nl/8610/1/Al-MHMC-2009-M.VAN.VEEN.pdf					
<1% match (Internet from 17-Jan-2021) https://mobissoftinfotech.com/resources/blog/non-functional-testing-guide/					
<1% match (Internet from 06-Nov-2022) https://www.griin.com/free_download?document_id=279298&product_form=ebook&publication_type=pdf					
<1% match (Internet from 24-Sep-2022) https://fict.utar.edu.my/documents/FYP/IIPSPW_template/IIPSPW_Report_Template_CN.docx					
<1% match () Lancaster, Benjamin. "The Florida Project: A Micro-Budget Feature Comedy", "Information Bulletin on Variable Stars (IBVS)", 2016					
<1% match (Internet from 22-Jan-2024) http://suspace.su.edu.bd					
<1% match (Internet from 24-Apr-2023) https://testsigma.com/blog/non-functional-testing/					
<1% match (Internet from 16-Dec-2018) https://www.admysys.com/testing-qa-services/					
<1% match (Internet from 26-Sep-2020) https://www.answers.com/Q/Function_of_blood_vessels_in_the_eye					
<1% match (Internet from 30-Oct-2022) https://www.diva-portal.org/smash/get/diva2:945974/FULLTEXT02					
<1% match (Kotaiba Mokadam, Hiam Khoury. "AN AUTOMATED SEMANTIC SEGMENTATION-FREE APPROACH TO POINT CLOUD SCENE UNDERSTANDING IN CONSTRUCTION", Proceedings of the Creative Construction Conference 2023, 2023) Kotaiba Mokadam, Hiam Khoury. "AN AUTOMATED SEMANTIC SEGMENTATION-FREE APPROACH TO POINT CLOUD SCENE UNDERSTANDING IN CONSTRUCTION", Proceedings of the Creative Construction Conference 2023, 2023					
<1% match (publications) Vorderer Peter, Bryant Jennings. "Playing Video Games - ", Taylor & Francis, 2012					
<1% match (Internet from 16-May-2019) https://repository.au.edu/bitstream/handle/6623004553/17694/AU-Thesis-Fulltext-224006.pdf?isAllowed=y&sequence=8					

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1




FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	PANG QING SEN
ID Number(s)	21ACB06605
Programme / Course	BACHELOR OF COMPUTER SCIENCE (CS)
Title of Final Year Project	GAMIFICATION IN LEARNING OBESITY

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: <u> 3 </u> % Similarity by source Internet Sources: <u> 2 </u> % Publications: <u> 1 </u> % Student Papers: <u> 2 </u> %	
Number of individual sources listed of more than 3% similarity: <u> 0 </u>	
Parameters of originality required and limits approved by UTAR are as Follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.



 Signature of Supervisor

Name: Tong Dong Ling

Date: 12 Sep 2024

 Signature of Co-Supervisor

Name: _____

Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)**

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	21ACB06605
Student Name	Pang Qing Sen
Supervisor Name	Ts Tong Dong Ling

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 12/09/2024