

**SMELL PHISHY : A 2D PLATFORM GAME for
Phishing Awareness**

NG MAN TENG

**Bachelor of Computer Science (Graphics and
Multimedia Technology)**

**UNIVERSITI MALAYSIA PAHANG
UNIVERSITI MALAYSIA PAHANG**

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : NG MAN TENG

Date of Birth :

Title : SMELL PHISHY : 2D PLATFORM GAME ON PHISHING
AWARENESS

Academic Session : 2022/2023

I declare that this thesis is classified as:

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)

New IC/Passport Number
Date: 20 January 2022



(Supervisor's Signature)

Dr. Nor Saradatul Akmar Binti Zulkifli

Name of supervisor
Date: 27/2/2023

NOTE: * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
Perpustakaan Universiti Malaysia Pahang,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

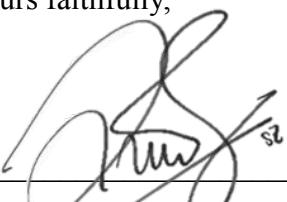
Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name	Ng Man Teng
Thesis Title	SMELL PHISHY: A 2D platform game for phishing awareness

Reasons	(i)
	(ii)
	(iii)

Thank you,

Yours faithfully,



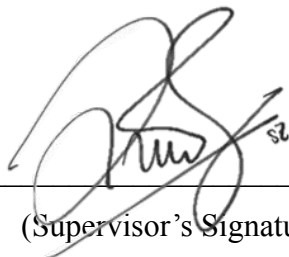
(Supervisor's Signature)

Date: 27/2/2023
Stamp: DR. NOR SARADATUL AKMAR BINTI ZULKIFLI
SENIOR LECTURER
FACULTY OF COMPUTING
COLLEGE OF COMPUTING & APPLIED SCIENCES
UNIVERSITI MALAYSIA PAHANG
26600 PEKAN, PAHANG DARUL MAKMUR
TEL: 09-424 4697 FAX: 09-424 4666

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

SUPERVISOR's DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Bachelor of Computer Science in Graphics and Multimedia Technology



(Supervisor's Signature)

Full Name : Ts. Dr. Nor Saradatul Akmar Binti Zulkifli
Position : Senior Lecturer
Date : 14/3/2022

(Co-supervisor's Signature)

Full Name :
Position :
Date :

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.



(Student's Signature)

Full Name : NG MAN TENG

ID Number : CD19101

Date : 14/3/2022

SMELL PHISHY 2D GAME: 2D PLATFORM PHISHING AWARENESS GAME

NG MAN TENG

Thesis submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Science (Graphic & Multimedia Technology)

Faculty of Computing
UNIVERSITI MALAYSIA PAHANG

JANUARY 2023

ACKNOWLEDGEMENTS

First of all, I want to say thank you to Universiti Malaysia Pahang and Faculty of Computing for providing support for equipment such as the library for doing research on the project and approving this project. Next, I would like to thank my project supervisor, Dr.Nor Saradatul Akmar Binti Zulkifli for providing support, ideas, and advice on this project while I was facing problems with my project. I also like to thank my family for mental support when I was struggling with some project. I also like to thank my friends that provide me with some ideas and technical support while I am facing coding problems during my development process. Finally, thank you to everyone who might have helped me indirectly and I did not mention during the development of Smell Phishy : 2D platform game. for phishing awareness.

ABSTRAK

Projek ini membincangkan pembinaan permainan platform Smell Phishy : 2D. Permainan ini adalah untuk kesedaran phishing data dan ia boleh dimainkan pada komputer atau komputer riba, dengan papan kekunci dan tetikus. Masalah projek adalah, terdapat pelbagai kaedah untuk mengajar kesedaran phishing tidak cekap. Jadi, permainan platform 2D boleh menjadi satu cara yang menarik dan ia membenarkan pemain melibatkan diri mereka di dalam permainan tersebut. Kelebihan permainan platform 2D tersebut adalah permainan platform 2D mempunyai interaksi dengan pemain. Ini membenarkan pemain memahami kepentingan kesedaran phishing dengan mudah dan seronok. Objektif projek ini adalah untuk mengkaji teknologi yang digunakan semasa pelaksanaan permainan platform 2D, untuk membangunkan permainan platform 2D untuk kesedaran phishing, dan untuk menilai fungsi permainan platform 2D yang dibina untuk kesedaran phishing. Metodologi yang dipilih ialah Pembinaan Aplikasi Rapid. Permainan ini dibina dengan menggunakan Kod Unity dan Visual Studio. Terdapat 20 respons terhadap UAT(User Acceptance Test) dan kebanyakan mereka menganggap permainan ini bagus tetapi masih perlu diperbaiki.

ABSTRACT

This project is discussing development of the Smell Phishy : 2d platform game for phishing awareness which can be played using a computer or laptop with keyboard and mouse. The problem statement of the project is that many methods to teach phishing awareness are inefficient. So, the 2D platform game can become an interesting way and can let players involve themselves inside the game. The 2D platform game can bring advantages which can have an interaction with the player. This can let players understand the importance of phishing awareness with fun and ease. The objective of this project is to study the technology behind the implementation of a 2D platform game, to design and develop a 2D platform game for phishing awareness, and to evaluate the functionality of the developed 2D platform game for the phishing awareness. The methodology selected is Rapid Application Development. The game was developed by using Unity and Visual Studio Code. There are 20 responses to the User Acceptance Test and most of them think the game is good but still need to improve.

TABLE OF CONTENT

ACKNOWLEDGEMENTS	7
ABSTRAK	8
ABSTRACT	9
TABLE OF CONTENT	10-12
LIST OF TABLES	12
LIST OF FIGURES	13-18
CHAPTER 1 INTRODUCTION	
1.1 INTRODUCTION	19-20
1.2 PROBLEM STATEMENTS	21-22
1.3 OBJECTIVE	23
1.4 SCOPE	23
1.5 SIGNIFICANCE OF PROJECT	24
1.6 REPORT ORGANIZATION	24
CHAPTER 2 LITERATURE REVIEW	
2.1 INTRODUCTION	25
2.2 PHISHING ATTACK AWARENESS	25
2.3 REVIEW OF EXISTING SYSTEM	26
2.3.1 MEGA MAN X	26-28
2.3.2 SUPER MARIO BROS	29-31
2.3.3 METAL SLUG	32-35
2.3.4 ANTI-PHISHING PHIL	36-39
2.3.5 KEEP TRADITION SECURE	39-41
2.3.6 TARGETED ATTACK: THE GAME	42-43
2.4 COMPARISON OF EXISTING SYSTEMS	
2.4.1 COMPARISON OF EXISTING SYSTEMS	44-45
2.4.2 ADVANTAGES AND DISADVANTAGES OF EXISTING SYSTEM	46-47
2.5 SUMMARY OF REVIEW EXISTING SYSTEM	48-49
CHAPTER 3 METHODOLOGY	

3.1	INTRODUCTION	50-51
3.2	PHASE 1 - ANALYSIS QUICK DESIGN	51
3.2.1	ANALYSIS OF EXISTING GAMES	52
3.2.2	PROJECT REQUIREMENT	52
3.2.3	FUNCTION AND NON-FUNCTIONAL REQUIREMENT	53
3.2.4	CONSTRAINTS AND LIMITATIONS	54
3.2.5	CONTEXT DIAGRAM	55
3.2.6	USE CASE DIAGRAM	56
3.2.7	ACTIVITY DIAGRAM	57-58
3.2.8	GAME DESIGN	
3.2.8.1	GAME MECHANICS	59
3.2.8.2	PLAYER CONTROL	59
3.2.8.3	GAME CHARACTER	
A.	MAIN CHARACTER	59
B.	NON PLAYER CHARACTER	60
C.	ENEMY	60
3.2.9	LEVEL DESIGN	60
3.2.10	REWARD AND PUNISHMENT	61
A.	REWARD	61
B.	PUNISHMENT	61
3.2.11	CHALLENGES AND OBJECTIVES	61
3.2.12	STORYBOARD	62-70
3.3	PHASE 2 - PROTOTYPING	71
3.3.1	BUILD PROTOTYPE	71
3.3.2	DEMONSTRATE	71
A.	DEMONSTRATE PROTOTYPE	71
B.	GATHER FEEDBACKS	71
3.3.3	REFINE	71
A.	UPDATE THE REQUIREMENT OF PROPOSED GAME	71
B.	FIX THE BUG	72-73
3.4	PHASE 3 - TESTING	73
A.	USER ACCEPTANCE TEST	74
3.5	PHASE 4 - IMPLEMENTATION	74
3.6	CONCLUSION	75

CHAPTER 4 DEVELOPMENT PROGRESS

4.1	INTRODUCTION	75
4.2	PROTOTYPE CYCLES PROCESS	75-76
4.2.1	BUILD	76-77
A.	DEVELOPMENT TOOLS	77
B.	DESIGNING GRAPHIC AND MULTIMEDIA CONTENTS	77-80
C.	LEVEL DESIGN	81
D.	DEVELOPMENT OF THE GAME APPLICATION	81
E.	USER INTERFACE	82-103
F.	GAME MECHANIC	104-136
4.2.2	DEMONSTRATE	137
A.	DEMONSTRATE PROTOTYPE	137
B.	GAIN FEEDBACK	137
4.2.3	REFINE	137
A.	UPDATE REQUIREMENT OF PROPOSED GAME	137
B.	FIX THE BUG	138-141
4.3	TESTING (UAT)	142-157
4.4	IMPLEMENTATION	158

CHAPTER 4 DEVELOPMENT PROGRESS

5.1	OBJECTIVE REVISITED	159
5.2	LIMITATION	159-160
5.3	FUTURE WORK	160

REFERENCES	161-162
-------------------	----------------

APPENDIXES	163-229
-------------------	----------------

LIST OF TABLES

Table 2.1: Comparison of existing 2D platform games	44
Table 2.2: Comparison of existing cybersecurity awareness games	45
Table 2.3: Advantages and Disadvantages of existing 2D platform games.	46
Table 2.4: Advantages and disadvantages of existing cybersecurity awareness games.	47
Table 3.0 The functional and non-functional requirements	53
Table 3.1 Constraints and Limitations	54
Table 3.2 : Storyboard of main menu	62
Table 3.3 : Storyboard of Game UI	64
Table 3.4 : Storyboard of Credits UI	65
Table 3.5 : Storyboard of Level UI	66
Table 3.6 : Storyboard of Game Complete UI	67
Table 3.7 : Storyboard of Pause menu UI	68
Table 3.8 : Storyboard of health bar of player character	69
Table 3.9 : Storyboard of ladder	69
Table 3.10 : Storyboard of player character with the coin	70
Table 3.11 : Storyboard of question about phishing	70
Table 3.12: Requirement form	72
Table 3.13: Template of Functional acceptance testing	73
Table 4.2.1: Prototype cycles process	75
Table 4.2.2 : Tools used for the development of game application	77
Table 4.2.2 : Level design	81
Table 4.2.3.1: Requirement form	138

LIST OF FIGURES

Figure 2.1 : 2D platform game name “Mega man X”	26
Figure 2.2 : Game controls of the Mega Man X	27
Figure 2.3 : Player character shooting the enemy	27
Figure 2.4 : Wall slide in Mega Man X	28
Figure 2.5 : The boss of the level in Mega Man X.	28
Figure 2.6: Screenshots of Super Mario Bros	29
Figure 2.7 : The mushroom pop out after the player character collide with the box	30
Figure 2.8 : The coin pops out when the player character collides with the box.	30
Figure 2.9 : The player character kills the enemy by jumping on top of the enemy.	30
Figure 2.10 : Game over UI pops out when the player character becomes 0.	31
Figure 2.11 : 2D platform game name “Metal Slug”	32
Figure 2.12 : The UI when the game starts.	33
Figure 2.13 : UI of soldier selection.	33
Figure 2.14 : UI of the game and the character shooting the enemy.	34
Figure 2.15: Player character gets a powerful weapon.	34
Figure 2.16 : Player character saves the victim and gets a reward.	35
Figure 2.17 : Game over UI	35
Figure 2.18 : Cover of Anti-Phishing Phil	36
Figure 2.19: the instruction of the game.	37
Figure 2.20: UI of the game and the fish is the player character	37
Figure 2.21: The player character gets the correct worm.	38
Figure 2.22: The player character gets the wrong worm and the answer.	38
Figure 2.23: The page explains the URL after round 1.	39
Figure 2.24: The tutorial before next round.	39
Figure 2.25: the cover of the keep tradition secure	40
Figure 2.26: the question about cybersecurity.	40
Figure 2.27: Page shows after answering the question correctly.	41
Figure 2.28 : Gotcha page after answering the question wrongly.	41
Figure 2.29: Opening video of the game	42
Figure 2.30: Selection page after the opening video	42
Figure 2.31: the decision page	43
Figure 3.0: The phases of RAD	50
Figure 3.1: The process of the RAD is used for game development.	51
Figure 3.2 : Context Diagram of Smell Phishy, 2D platform game	55
Figure 3.3 : Use Case Diagram of “Smell Phishy”	56
Figure 3.4 : Activity diagram of “Smell Phishy”	57
Figure 3.5 : Main Character of the game	59
Figure 3.6 : Non Player Character (NPC)	60
Figure 3.7 : Enemy	60

Figure 4.2.1.1 : Example of editing Npc in Adobe Photoshop	78
Figure 4.2.1.2: Example of editing image arrow as trap in Adobe Photoshop	78
Figure 4.2.1.3: Animation of enemy editing in Adobe photoshop	79
Figure 4.2.1.4 : Some example of 2D sprites that used in the game application	79
Figure 4.2.1.5 : Text “PressStart2P-Regular” used as the game UI	80
Figure 4.2.1.6 : Audio editing in Kawing	80
Figure 4.2.1.7 : Main Menu Scene	82
Figure 4.2.1.8 : Inspector of Start button	83
Figure 4.2.1.9: Inspector of level button	84
Figure 4.2.1.10: Inspector of Credit button	85
Figure 4.2.1.11 : Inspector of Quit button	85
Figure 4.2.1.12: Script of Main menu	86
Figure 4.2.1.13 : Confirmation message.	87
Figure 4.2.1.14: Inspector of “Yes” button in confirmation page	88
Figure 4.2.1.15: Inspector of “NO” button.	88
Figure 4.2.1.16: Credits page.	89
Figure 4.2.1.17: Inspector of back button.	89
Figure 4.2.1.18: Level page	90
Figure 4.2.1.19: Inspector of the Level.	91
Figure 4.2.1.20: Mask Inspector	91
Figure 4.2.1.21: Inspector of the Reset button	92
Figure 4.2.1.22: Inspector of the Back button of level page	92
Figure 4.2.1.23: Inspector of the level management.	93
Figure 4.2.1.24: LevelSelection Script.	93
Figure 4.2.25: Pause menu page	94
Figure 4.2.1.26: Pause Menu Script.	95
Figure 4.2.1.27: Game complete.	96
Figure 4.2.1.28: Interface of the tutorial	97
Figure 4.2.1.29: Dialog of the NPC	97
Figure 4.2.1.30: Inspector of Signbox.	98
Figure 4.2.1.31: Script of Sign.	99
Figure 4.2.1.32: Script of SignNotDestroy	99
Figure 4.2.1.33: Loading Scene	100
Figure 4.2.1.34: Script of Loading Scene.	101
Figure 4.2.1.35: Game over panel	102
Figure 4.2.1.36: Inspector of the game over panel	102
Figure 4.2.1.37: Insufficient coins panel	103
Figure 4.2.1.38: Inspector of the insufficient coins panel.	103
Figure 4.2.1.39: Phishing Awareness Question Page	104
Figure 4.2.1.40: After answering the question wrongly.	105
Figure 4.2.1.41: After answering the question correctly.	105

Figure 4.2.1.42: The question panel disappears.	106
Figure 4.2.1.43: The computer disappears.	106
Figure 4.2.1.44: The inspector of Quiz Manager.	107
Figure 4.2.1.45: Script of Quiz Answer and Question.	107
Figure 4.2.1.46: Script of Answer	108
Figure 4.2.1.47: Script of Quiz Manager.	109
Figure 4.2.1.49: Inspector of computer	110
Figure 4.2.1.50: Script of pop out.	111
Figure 4.2.1.51: Enemies and traps in tutorial and Level 2.	112
Figure 4.2.1.52: Enemies and traps in Level 3 and Level 4.	113
Figure 4.2.1.53: Enemies and traps in Level 5 and 6.	113
Figure 4.2.1.54: Enemies and traps move following the waypoints.	113
Figure 4.2.1.55: Inspector of enemies and traps that can move.	114
Figure 4.2.1.56: Script of Enemy	114
Figure 4.2.1.57: Script of Enemy AI	115
Figure 4.2.1.58: Traps without moving.	115
Figure 4.2.1.59: Inspector of the traps	116
Figure 4.2.1.60: Script of traps.	116
Figure 4.2.1.61: Virus enemy	117
Figure 4.2.1.62: Inspector of virus enemy	118
Figure 4.2.1.63: Script of Chase Player	119
Figure 4.2.1.64: Script of reset position.	120
Figure 4.2.1.65: Arrow trap.	120
Figure 4.2.1.66: Inspector of arrow traps.	121
Figure 4.2.1.67: Script of Arrow Trap.	121
Figure 4.2.1.68: Inspector of arrow.	122
Figure 4.2.1.69: Script of Enemy projectile.	123
Figure 4.2.1.70: Falling platform.	123
Figure 4.2.1.71: Script of falling platform	124
Figure 4.2.1.71: Rotating cube.	124
Figure 4.2.1.72. Inspector of cubes.	124
Figure 4.2.1.73: The moving platform and ladder	125
Figure 4.2.1.74: Three box collider 2D and inspector.	125
Figure 4.2.1.75: Inspector of moving platform	126
Figure 4.2.1.76: Script of waypoint.	126
Figure 4.2.1.77: Inspector of player character.	127
Figure 4.2.1.78: Script of Climb.	128
Figure 4.2.1.80: Script of Player movement.	129
Figure 4.2.1.81: Script of Player Life	130
Figure 4.2.1.82: Script of Health Bar	130
Figure 4.2.1.83: Script of item collect	131

Figure 4.2.1.84: Coins, and Checkpoint.	131
Figure 4.2.1.85: Inspector of coins	132
Figure 4.2.1.86: Inspector of checkpoint	133
Figure 4.2.1.87: Script of Checkpoint tutorial.	134
Figure 4.2.1.88: Script of Not Enough Coin	134
Figure 4.2.1.89: Inspector of Camera	135
Figure 4.2.1.90: Script of Camera Controller	135
Figure 4.2.1.91: NPC	135
Figure 4.2.1.92: Inspector of NPC.	136
Figure 4.2.1.93: Script of NPC	136
Figure 4.3.1: Result for the Institution.	142
Figure 4.3.2: Result of Institution name.	142
Figure 4.3.3: Result of Role of survey	143
Figure 4.3.4: Result of gender.	143
Figure 4.3.5: Result of age of survey.	144
Figure 4.3.6: Result of Have You Studied Phishing Awareness.	144
Figure 4.3.7: Result of the rate of main menu function	145
Figure 4.3.8: Rate of the Level page function.	145
Figure 4.3.9: Rate of function of the Credits page.	146
Figure 4.3.10: Rate of the function of the Question page.	147
Figure 4.3.11: Rate of the function of player character	147
Figure 4.3.12: Rate of the function of the NPC.	148
Figure 4.3.13: Rate of the function of the question panel.	148
Figure 4.3.14: Rate of the function of the health bar of the player character.	149
Figure 4.3.15: Rate of function of coin.	149
Figure 4.3.16: Rate of function of the enemies and traps.	150
Figure 4.3.17: Rate of function of light and neon system.	151
Figure 4.3.18:Rate of the function of the moving platform.	151
Figure 4.3.19: Rate of the function of the falling platform.	152
Figure 4.3.20: Rate of the function of the checkpoint.	152
Figure 4.3.21: Rate of function of ladder.	153
Figure 4.3.22: Comment of the function of the game.	154
Figure 4.3.23: Rate of the likes of the game.	154
Figure 4.3.24: Rate of game enjoyment.	155
Figure 4.3.25: Rate of did player gain knowledge from the game.	155
Figure 4.3.26: Rate of Do players want to play the game again.	156
Figure 4.3.27: Rate of marketability of the game.	156
Figure 4.3.28: Comments of the game.	157

LIST OF APPENDIXES

Appendix 1: User Acceptance Testing Form	163-166
Appendix 2: Script of Main menu	167-168
Appendix 3: Script of Level Selection	169
Appendix 4: Script of Pause menu	170
Appendix 5: Script of Quiz answer and question	171
Appendix 6: Script of Answer	171
Appendix 7: Script of Quiz manager	172-173
Appendix 8: Script of Sign	174
Appendix 9: Script of Pop out	175
Appendix 10: Script of Waypoint	176
Appendix 11: Script of Enemy	176
Appendix 12: Script of Enemy AI	177-178
Appendix 13: Script of Trap Object	179
Appendix 14: Script of Chase Player	180-181
Appendix 15: Script of Reset position	182
Appendix 16: Script of Arrow trap	183
Appendix 17: Script of Enemy projectile	184
Appendix 18: Script of Falling platform	185
Appendix 19: Script of Climb	186
Appendix 20: Script of Sticky platform	187
Appendix 21: Script of Player Movement	188-189
Appendix 22: Script of Player life	190-191
Appendix 23: Script of Health bar	192
Appendix 24: Script of Item collect	193
Appendix 25: Script of Checkpoint tutorial	194-195
Appendix 26: Script of Not enough coin	195
Appendix 27: Script of Camera Controller	196
Appendix 28: Script of NPC	197
Appendix 29: Script of Sign Not destroy	198
Appendix 30: Script of Loading Scene	199
Appendix 31: Table Function Acceptance Test	200 - 221
Appendix 32: User Acceptance Test	222 - 228
Appendix 33: Page in Itchi.io and QR code to download	229

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In this digital world, there are many phishing attacks happening in this world. This is because the hackers will try to gain access to any account by trying to let the people click in the link to install malware into the computer. After the malware is installed, they are able to access the device and gain the information they want. There are also many types of the phishing attack such as email phishing, vishing (voice phishing), Spear phishing and Smishing (SMS phishing). There are many conventional methods to spread the phishing awareness such as having a talk, some interactive video or powerpoint and quizzes. However, spreading the phishing awareness by using a 2D platform game will be more interesting and players can learn and remember the phishing awareness in a fun way.

2D platform games are a video game genre. It is a game that needs the player to move the character to the destination and move the character to avoid the obstacles. Players also can use the character to fight or shoot the enemy to get the rewards or reach their goal to complete the level. 2D platform games originated in the early 1980s. There is a new arcade game created by Nintendo and named it "Donkey Kong". It was released in July 1981 and it was the first game to allow players to jump to avoid obstacles. and it became the first popular game being 1982's Pitfall Years. The 2D platform game David Crane's Pitfall became one of the best-selling games. Donkey Kong was ported to numerous comforts and computers at the same time and system-selling-pack-in for ColecoVision (second generation home video game console) (Jmelicantafe, James Melican, 2017). The game "Donkey Kong Jr" and "Mario Bros" are the first 2D platform game with two-player cooperative play. (Jmelicantafe, James Melican, 2017).

2D platform games have been applied in the education field to create an interesting, unique and attractive education learning for the people. 2D platform games can become one of the methods for teaching and learning. 2D platform games can be used as a tool to improve the learning experience of learners and also learn some skills such as following the rules of the game, interaction, creativity, teamwork and so on. This can make

the learner participants inside the game and learn easily compared to the traditional teaching methods. Bringing 2D platform games in teaching can let learners learn and understand the things easily with fun.

Phishing awareness is very important to prevent phishing attacks from happening in this digital world. This is because they can get the important information by accessing the computer after the victim clicks on the link and install the malware in the victim's computer. For example, they can get the password and number of bank accounts to transfer money to their account. This will cause the money to be gone and important information leaked.

So, the proposal of the 2D game “Smell Phishy” is to let the user gain knowledge on phishing awareness and alert on the phishing attack. There are many levels and the difficulty of the game will increase level by level. In the game, the player will need to control the character to avoid colliding with the obstacle or enemy and answer some questions about the phishing awareness. By answering the question, players can understand phishing awareness and remember the need to be alert on the spam email, SMS, and unknown telephone call. The reason for using games to teach players about phishing awareness is that most people love to play some interesting and fun games to relax themselves compared to learning from studying. So, games are a way to teach people about phishing awareness from fun and they will more easily remember and understand the knowledge compared to studying.

1.2 PROBLEM STATEMENTS

Nowadays, a smartphone has become an essential thing for everyone and a computer or a laptop also became an important item for the student to study. All of it has brought many advantages for the people in this digital world. For example, people can contact each other by using a smartphone or computer and transfer the money by scanning the Qr code or having the bank account of the receiver. This makes the life of people easier and convenient compared to the old generation. However, there also have many disadvantages such as the phishing attack will happen. In this digital generation, a game will be more effective to teach people about the importance of phishing awareness.

There are many existing ways to learn about phishing awareness and joining a talk is one of the ways to learn about phishing attacks. An expert speaker can speak out about their experience and important notes they learn in their life. This is a way to share profound knowledge on phishing awareness to the people. However, still many people do not alert the importance of phishing awareness and this causes them not interested in joining the talk. They also have phishing simulation training that is carried out by the experienced teacher to teach the people how to use the best way to solve the phishing attack. However, the people also lack the knowledge about the importance of the phishing attack and they lack interest in the training if they need to pay for it.

There are also interactive video and powerpoint presentations about phishing awareness. This was making the teaching method more interesting compared to the traditional teaching method. People will watch the video to gain knowledge. However, it is not really effective compared to using a game. People will not remember or understand the video easily and need to repeat to watch the video until they understand the video.

There are also many quizzes about phishing awareness. The people can learn the importance of phishing awareness by answering the quizzes. However, the people were not aware of the importance of the phishing awareness and not interested in answering the quizzes even though the quizzes are free for the people to answer.

Therefore, there needs to be a method to solve the problem to let the people understand the importance of the phishing awareness and remember the phishing awareness

in an easy way with fun. An educational game can provide effective teaching because it is attractive and motivated. So, this project decided to develop a 2D platform game to raise the phishing awareness of the people.

1.3 OBJECTIVE

There are three objectives in this project, which are:

- To study the existing 2D platform game and game about phishing awareness.
- To design and develop a 2D platform game for phishing awareness.
- To evaluate the functionality of the developed 2D platform game for the phishing awareness.

1.4 SCOPE

The scopes of “Smell Phishy” 2D platform game are as follows:

User Scope:

- I. This game is available for players above 18 years old and have knowledge about phishing awareness.

System Scope:

- I. This game is a 2D platform game.
- II. This game is available for pc and it is an offline game. .
- III. This game only supports single player mode.
- IV. The player needs to control the character by using the keyboard and mouse.
- V. The player can control the character to collect the coins in the game.
- VI. The player can control the character to avoid the obstacles and enemy by pressing the jump button to let the character jump and jump on top of the enemy to kill them.
- VII. There are some questions about phishing awareness pop out in the game and money will be lost when players answer it incorrectly.
- VIII. The player needs to complete the level by reaching the objective in the level of the game.
- IX. There are 6 levels in the game.

Development Scope:

- I. This game is using Unity Software to develop it.
- II. This game contains multimedia elements such as 2D animation, text, music, sound and graphics.

1.5 SIGNIFICANCE OF PROJECT

I. People

The people can have a process of learning in an interactive way and gain the phishing awareness easily compared to the transitional teaching. They can learn about phishing awareness in an interesting way and help them to avoid phishing attacks.

1.6 REPORT ORGANIZATION

This report contains five chapters. Chapter 1 is about the overview of the project including introduction, problem statements, objective, scope, significance of project.

Chapter 2 is literature review on 3 existing 2D platform games that were available in the market. These 3 existing 2D platform games and 3 existing cybersecurity awareness games will be used to do comparison with the proposed application. The comparison between the proposed application and 3 existing games and 3 existing cybersecurity awareness games based on the platform that supports the game, the features of the game, weaknesses of the game, advantages and disadvantages of the game and the way to teach cybersecurity awareness and the way to perform in the game. .

Chapter 3 is methodology and it is used to develop the application. This chapter discusses the design and interface of the game. This chapter also discusses the hardware and software specification of the game and the flow of the game.

Chapter 4 explains the development process and implementation of the project. All the output and the result of the project were discussed in this chapter. Software development, testing, data collection and the result of the project also were explained in this chapter.

Chapter 5 is about the conclusion of the project and summarizes the result of the project. The limitation and the further works also were discussed in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

Chapter 2 is about the review on the existing game applications for this project. In this project, the three existing 2D platform games and three existing cybersecurity awareness games were explained and compared. Advantages and disadvantages of the three existing 2D platform games and three cybersecurity awareness games also were discussed in this chapter. The comparison of the existing games are used to ensure this project will be developed in a better version.

2.2 PHISHING ATTACK AWARENESS

There are many different types of phishing attacks such as vishing(Voice phishing), email phishing, and smishing(SMS phishing). Vishing is attacked by using voice call, email phishing is really common and it uses spam email to attack the victim, and Smishing is attacked by sending SMS to the receiver. The phishing attack will happen because the people have a lack of phishing awareness. There are also many existing ways to teach phishing awareness such as interaction video, powerpoint, phishing simulation training, a talk and so on. However, all of them are not really effective and cause an increase in the case of phishing attacks. After playing some awareness games, the awareness game inspired me to make a 2D platform game to spread phishing awareness. This method can let players play, learn and test their phishing awareness knowledge in a fun and interesting way.

2.3 REVIEW OF EXISTING SYSTEMS

There are three existing games in 2D platform that will be reviewed which are Mega Man X, Super Mario Bros, and Metal Slug. There are three existing games in cybersecurity awareness that also will be reviewed which are Anti-Phishing Phil, Keep Tradition Secure, and Targeted Attack: The Game. This existing game application will be a research and help to develop this project.

2.3.1 Application 1 - Mega Man X

Figure 2.1 shows the cover of a 2D platform game named “Mega Man X”. Mega Man X is a 2D platform game and it is a run-and -gun gameplay known as Rockman X.



Figure 2.1 : 2D platform game name “Mega man X”

It is the first game of the Mega Man X series and was released from 1993-1995. This game is available on PC. It was developed and published by Capcom. In the game, players need to kill the enemy to increase and jump to avoid the obstacles. If the life of the character becomes 0, players need to restart the level. Players also will go to many different game environments to save the world by fighting with the enemy and boss of every level.

Mega Man X is a 2D platform game and Figure 2.2 shows the game controls of Mega Man X are simple to let the player understand. There are some basic movements by pressing the basic movement to move left, right, backward and forward. The character also can shoot the enemy by pressing the “F” button. Players also can change the game control to make them have a better game experience.

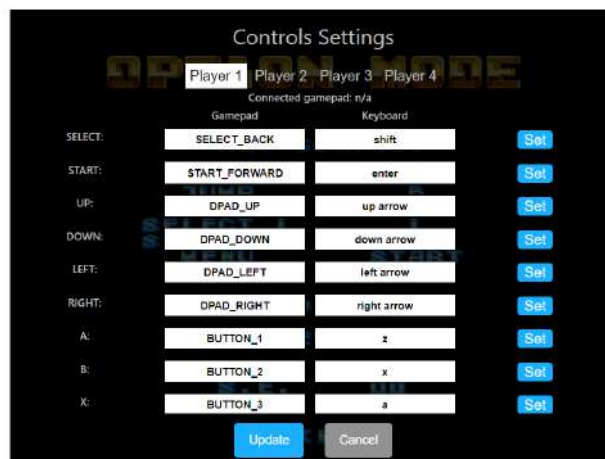


Figure 2.2 : Game controls of the Mega Man X

Figure 2.3 shows the player character shooting the enemy. The player character needs to kill the enemy by shooting to pass through the level. There are also many types of enemy in each level and the enemy can fly, walk or roll.



Figure 2.3 : Player character shooting the enemy

Figure 2.4 shows the player character is wall slide animation when the player character jumps near the wall. The player character will fall slowly and the player character can jump again to move up.



Figure 2.4 : Wall slide in Mega Man X

The figure 2.5 shows the boss of the level. There are many enemies in the game and this is one of the bosses in the level. Players need to kill the enemy to pass the level. If the health bar of the character becomes 0, the player character will play the dead animation and the player needs to restart the level. Player needs to be alert to avoid colliding with the enemy and kill them by shooting.



Figure 2.5 : The boss of the level in Mega Man X.

2.3.2 Application 2 - Super Mario Bros

The figure 2.6 is a screenshot of Super Mario Bros and the instruction to teach the player how to control the player character when the game started. Super Mario Bros is a platform game developed and published by Nintendo. It was released in 1983 and became the first game in the Super Mario series. This game is available on PC and Nintendo. The UI of the game is simple and easy to understand.



Figure 2.6: Screenshots of Super Mario Bros

The game also includes some special bricks marked with questions as shown in Figure 2.7 and 2.8. This will pop out some coins or special items. The figure 2.7 shows the mushroom pop out when the player character collides with the special bricks. The character of the game is named Mario and the player needs to control this character to collect the mushroom and let the character become bigger to get a free life. The figure 2.8 also shows the coin popping out when the character collides with the special brick. The coin will be a reward and the number of the coins will increase that show in figure 2.8.



Figure 2.7 : The mushroom pop out after the player character collide with the box



Figure 2.8 : The coin pops out when the player character collides with the box.

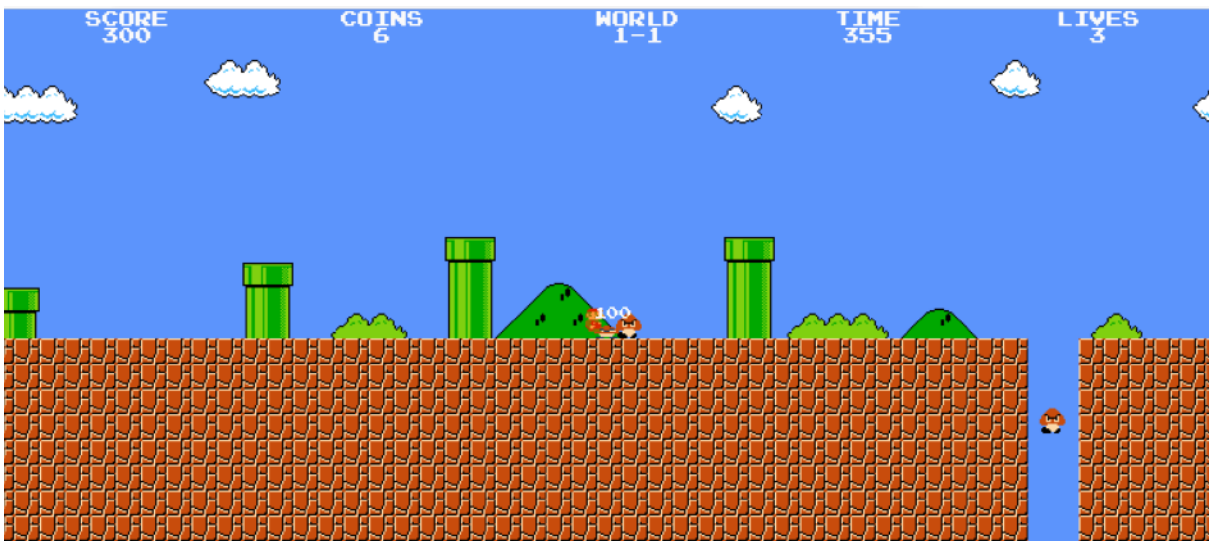


Figure 2.9 : The player character kills the enemy by jumping on top of the enemy.

The figure 2.9 shows the player character killing the enemy by jumping on top of the enemy. Players can jump though the enemy but the score would not increase. So, the benefit of

killing an enemy can increase the score and the score will be the reward. Players also need to jump and walk to avoid the obstacles.



Figure 2.10 : Game over UI pops out when the player character becomes 0.

Figure 2.10 shows the game over UI. The game over UI will pop out when the player character becomes 0. After a few seconds, the player will need to replay the level.

2.3.3 Application 3 - Metal Slug

The figure 2.11 shows the cover of the 2D platform game named “Metal Slug”. Metal slug is a 2D platform game and it is a Japanese run-and-gun video game developed by Nazca Corporation in 1996. This game is available on PC and PlayStation.



Figure 2.11 : 2D platform game name “Metal Slug”

The figure 2.12 shows the start UI of the game. The player can choose the arcade mission to start the game and combat school for tutorial of combat skill. The art gallery is used to show the model of the character that is inside the game and the option is used to show the game control of the game. The UI is simple and clean to let players feel comfortable.

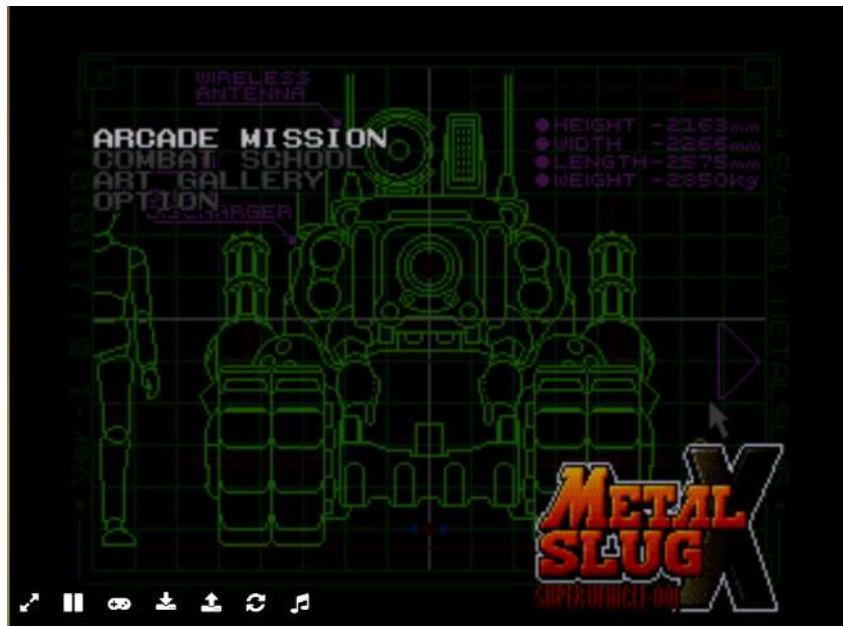


Figure 2.12 : The UI when the game starts.

Figure 2.13 shows 4 characters and one of them to become a player character and start the game. The player can choose one of the characters who wants to destroy the governments and set up a new order and all of the character's family are dead because the Regular Army was corrupt as shown in Figure 2.13 .



Figure 2.13 : UI of soldier selection.

Figure 2.14 shows the UI of the game and there is a player character shooting the enemy, The UI of the game is simple and can let players easily understand. The “2500” that

show in the figure 2.14 is the score of the game, the arms is the ammo of gun, bomb is the number of bomb, the “57” show in the figure 2.14 is the time left of the game and the “1UP” is the power of the weapon. The number of “1UP” will increase as shown in the figure 2.15.



Figure 2.14 : UI of the game and the character shooting the enemy.



Figure 2.15: Player character gets a powerful weapon.

Figure 2.16 shows the player character saving the victim and getting a reward from the victim to increase the score. On the way to the game, players also can save the victim in the game to receive some rewards to increase the score or get a powerful weapon.



Figure 2.16 : Player character saves the victim and gets a reward.

Figure 2.17 shows the game over UI. The player can press any button to continue the game or wait for the countdown to become 0 and the player needs to restart the level of the game.



Figure 2.17 : Game over UI

2.3.4 Application - Anti-Phishing Phil

The figure 2.18 shows the cover of the 2D game about phishing awareness named “Anti-Phishing Phil”. Anti-Phishing Phil is a 2D phishing awareness game and it is an easy game for employees and customers on how to spot phishing attacks.



Figure 2.18 : Cover of Anti-Phishing Phil

The figure 2.19 shows the instruction of the game to let players understand the rules of the game before starting the game. The player character is a fish named “Phil”. The player can control the player character by using the mouse. There are also some worms as food for the player character. However, players need to alert the worms because some of it will be a trap. So, players need to differentiate the URL that shows after the player character near to the worms. The player can press “E” to eat the worm or “R” to reject the worm or “T” to teach the father fish to differentiate if the URL is legitimate or fake. There also will be some enemies swimming in the bay and players to avoid them to prevent losing life.

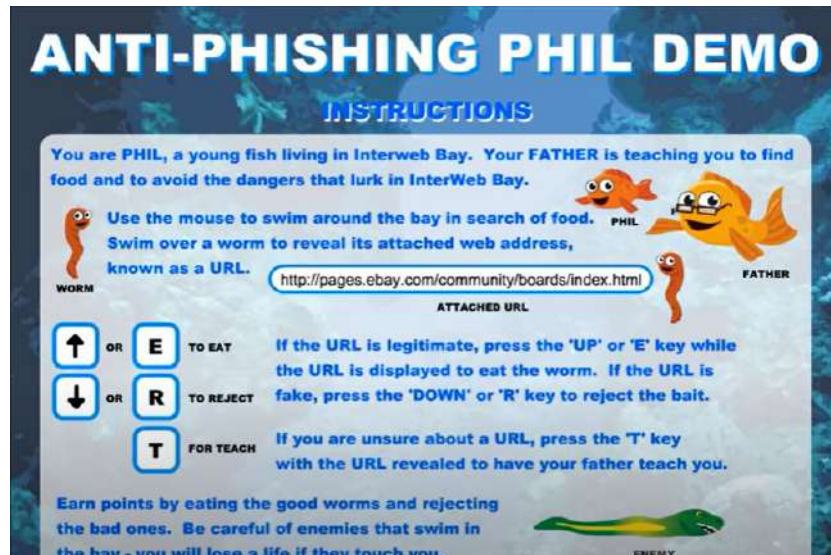


Figure 2.19: the instruction of the game.

The figure 2.20 shows the URL after the player character which is the fish “Phil” collides with the worms. The player needs to differentiate whether the URL is a legitimate or fake URL to earn the score and win the game.



Figure 2.20: UI of the game and the fish is the player character

The figure 2.21 shows the player character gets the correct worm with the legitimate URL. When the player gets the legitimate URL, the score of the player will increase and the game also will show a word “Yummy” to let the player know the answer is correct.

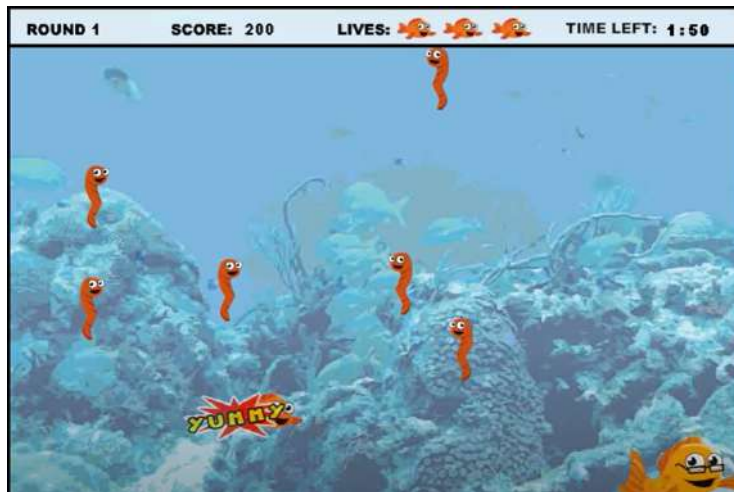


Figure 2.21: The player character gets the correct worm.

The figure 2.22 shows the player character get the wrong answer and lose the life of the player character. After that, the father fish will teach the player why it is the fake URL. The UI of the game also deduct the life of the player character.

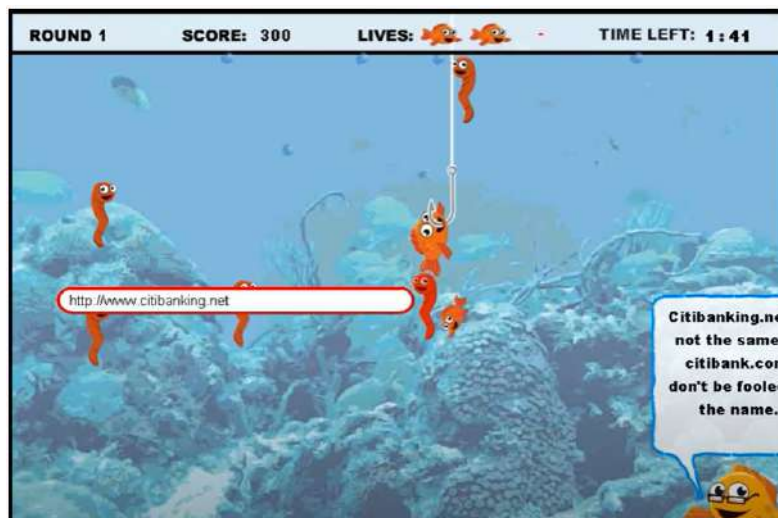


Figure 2.22: The player character gets the wrong worm and the answer.

The figure 2.23 shows the page that explains the reason for the legitimate URL and fake URL after one round. This can help players understand and know how to avoid the phishing attack by getting knowledge about the legitimate and fake URL.



Figure 2.23: The page explains the URL after round 1.

The figure 2.24 shows the tutorial page before the next round. This can let players gain some knowledge about phishing awareness and become alert on the URL.



Figure 2.24: The tutorial before next round.

2.3.5 Application 5 - Keep Tradition Secure

The figure 2.25 shows the cover of the game named “Keep Tradition Secure”. The game is about a hacker named “Bad Bull” who is threatening Texas A&M’s campus traditions and tracking this threat monger down needs to answer a series of cybersecurity questions.

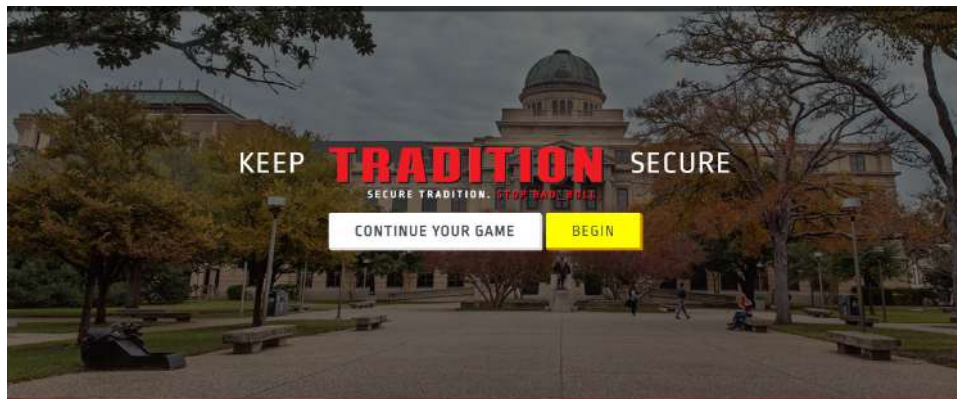


Figure 2.25: the cover of the keep tradition secure

The figure 2.26 shows the question about cybersecurity and players need to answer the question correctly. Players need to answer the question correctly based on the knowledge that they learned before.

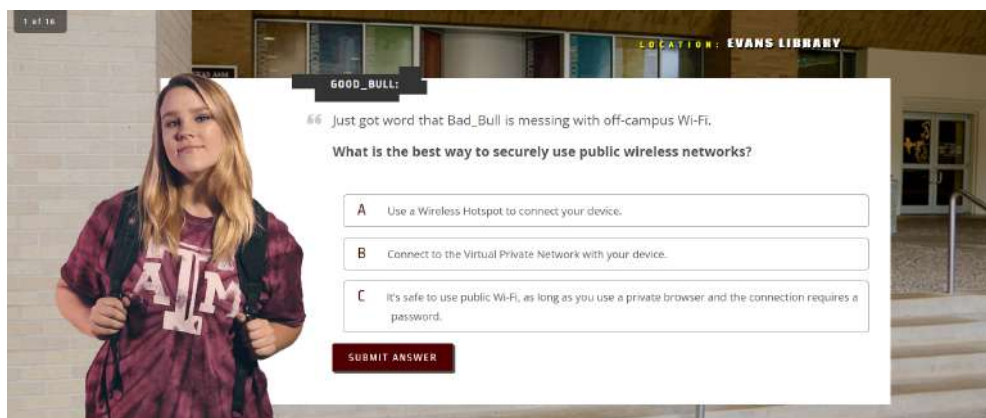


Figure 2.26: the question about cybersecurity.

The figure 2.27 shows the explanation page after the player answers the question correctly. The character in the game will show the congratulatory pose and explain the reason for the answer.



Figure 2.27: Page shows after answering the question correctly.

The figure 2.28 shows the gotcha page after the player answers the question wrongly. This figure shows that the computer has installed the malware and the computer has been hacked and the important information of the computer will be lost.



Figure 2.28 : Gotcha page after answering the question wrongly.

2.3.6 Application 6 - Targeted Attack: The Game

“Targeted attack: the game” is a game about the company to be always alert and keep aware of the environment. So, the boss can protect the information of the company. The figure 2.29 shows the opening video of the game to let players understand the storyline of the game before the game starts. The bottom left corner shows the subtitles. It can let the player understand the video by showing the subtitles and the player also can skip sections.

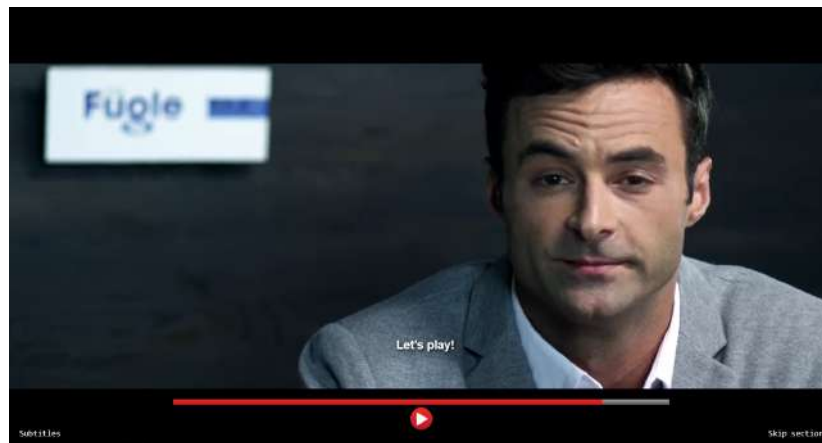


Figure 2.29: Opening video of the game

The figure 2.30 shows the selection page of the game after the video. The player can select the choice to decide to protect sensitive company information in light of potential security issues. The decisions that are made by the player are important because the decision can protect the important information of the company.



Figure 2.30: Selection page after the opening video

The figure 2.31 shows the decision page to make a decision to be prepared from the phishing attack. The top left corner shows the red coins used to choose to install the anti-virus application or the effective way to be used. The more effective way or the higher quality of the application install, more red coins will be used.

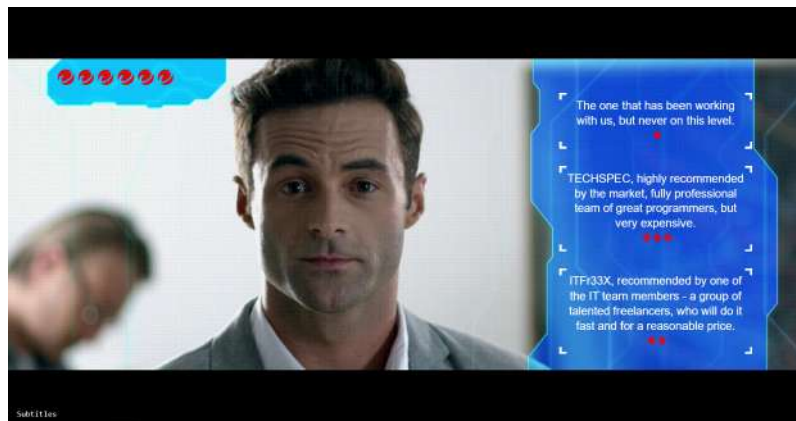


Figure 2.31: the decision page

2.4 COMPARISON OF EXISTING SYSTEMS

2.4.1 Comparison of existing systems

2.4.1.1 Comparison of existing 2D platform games

The specifications of the three existing 2D platform games are depicted in Table 2.1.

Table 2.1: Comparison of existing 2D platform games

Specification/ Features	Related work / existing systems		
	Mega Man X	Super Mario Bros	Metal Slug
Platform	PC	PC , Nintendo	PC, PlayStation
Graphical User Interface (GUI)	The interface is simple and well organised. However, the UI of the game does not show the score of the character.	The interface is simple and well organised.	The interface is simple and well organised.
Operating System (OS)	Windows, Android, IOS	Windows, Android, IOS	Windows, Android, IOS
Game Genre	Platform	Platform	Platform
Type of Connection	Offline	Offline	Offline
Game Type	2D	2D	2D
Control	Controller / Keyboard	Controller / Keyboard	Controller / Keyboard
Game Engine	Mega Engine	Unreal Engine 4	Unity
Total Player	Single	Single	Single/ Multiple
Price	RM45.00	\$4.99 RM 21.06	RM19.00

2.4.1.2 Comparison of existing cybersecurity awareness game

The specifications of the three existing cybersecurity awareness games are depicted in Table 2.2.

Table 2.2: Comparison of existing cybersecurity awareness games

Specification/ Features	Related work / existing systems		
	Anti-Phishing Phil	Keep Tradition Secure	Targeted Attack: The Game
Platform	PC	PC	PC
Game Genre	2D game	Question answering game.	Making a decision game.
Control	Keyboard/ Mouse	Mouse	Mouse
The way to teach cybersecurity awareness	The player will need to control Phil to get the correct worm with the legitimate URL to prevent a phishing attack.	The player needs to answer the question about cybersecurity awareness in the game correctly to chase the hacker “Bad Bull”.	The player needs to help the player character to make the decision to protect the important information of the company.
Total player	Single	Single	Single

2.4.2 Advantages and disadvantages of existing systems

2.4.2.1 Advantages and disadvantages of existing 2D platform games

All the related works have their advantages and disadvantages as depicted in Table 2.3.

Table 2.3: Advantages and Disadvantages of existing 2D platform games.

Related Works	Advantages	Disadvantages
Mega Man X	Classic design and interface Beautiful graphics and different environments	Monotonous gameplay UI do not show the score of the player character. Lack of game tutorial. Expensive on steam.
Super Mario Bros	Classic design and interface Cheap pricing	Monotonous gameplay No storyline
Metal Slug	Classic design and interface Cheap pricing Beautiful graphics and different environments Interesting gameplay Can play with other player	The UI of the game did not show the number of lives of characters. The player will lose the powerful weapon easily after colliding with the enemy one time. This makes the game more difficult.

2.4.2.2 Advantages and disadvantages of existing 2D platform games

All the related works have their advantages and disadvantages as depicted in Table 2.4.

Table 2.4: Advantages and disadvantages of existing cybersecurity awareness games.

Related Works	Advantages	Disadvantages
Anti-Phishing Phil	Have tutorials and teach the knowledge about the way to spot the fake URL. Hint given when the player presses the “T” button. Have an explanation after choosing the worm with the URL. Interesting gameplay	No storyline
Keep Tradition Secure	Have answers and explanations after the player answers the question.	Monotonous gameplay
Targeted Attack: The Game	Have a video to show the storyline to let the player understand and make the decision for the player character.	Monotonous gameplay No explanation after the decision was made.

2.5 SUMMARY OF REVIEW EXISTING SYSTEMS

Table 2.1 shows the comparison of the three existing games which are Mega Man X, Super Mario Bros and Metal Slug. Table 2.2 shows the comparison of existing phishing games which are Anti-Phishing Phil, Keep Tradition Secure, and Targeted Attack: The Game. Table 2.3 also shows advantages and disadvantages of the existing games and table 2.4 shows the advantages and disadvantages of the existing cybersecurity awareness games.

A clean, simple and beautiful graphical User Interface (GUI) is very important for a game to attract the players and make the player easily understand. The graphical user interface (GUI) of the existing games are simple and well organised to let players understand it easily based on their previous game experience. The proposed 2D platform game of the project “Smell Phishy” will make a simple, clean, interesting and user friendly graphical user interface (GUI) to make the players easily understand and attract the players. The game genre of three existing games are 2D platform games and the proposed project will be a 2D platform game. All of the existing games can be played offline and the proposed game also will be an offline game. The three existing games are controlled with a keyboard or controller and the proposed game also will be played by using a keyboard and mouse. The game engine that was used by Mega Man X was Mega Engine, super mario bros was Unreal Engine 4 and Metal Slug was using Unity. The proposed 2D platform game also will use the Unity game engine to develop the game.

The way to teach phishing awareness is also very important and the way to implement it into the 2D platform game. So, table 2.2 shows the comparison of the three existing cybersecurity awareness games and this table is used to investigate the way to teach the phishing awareness in the game. The way to teach in the Anti-Phishing Phil is the player needs to control the player character to get the correct worm with the legitimate URL. The father of the player character will give some tutorial, explanation, and hint to let the player understand the way to spot the phishing attack. Keep Tradition Secure and Targeted Attack: The Game also are games to teach phishing awareness but the way to teach is different with Anti-Phishing Phil. The way to teach the phishing awareness in Keep Tradition Secure is a question answering game and explanation will come out after the player answer the question and the player need to make the decision for the player character in the Targeted Attack: The Game after watch the video to understand the story in Targeted Attack: The Game. The way

to teach phishing awareness in the proposed game will be like Anti-Phishing Phil and Keep Tradition Secure, the proposed game will let the player identify the URL and some questions about the way to prevent phishing attacks in the way of question answering.

Table 2.3 and 2.4 shows some advantages and disadvantages for each of the existing 2D platform games and cybersecurity awareness games. The graphics and environment of Mega Man X and Metal Slug are beautiful and interesting compared to Super Mario Bros. The proposed game will use different types of environments to make the game more interesting. The proposed game will insert some interesting gameplay such as the player character colliding something on the way to answer the question to increase the coin. The advantages of the Anti-Phishing Phil and Keep Tradition Secure are the explanation and the answer given to let the player understand. The proposed game will ask the player some questions about cybersecurity awareness and the explanation will be given after the player chooses the answer. The disadvantages of the game is monotonous gameplay and it is a common disadvantage of Mega Man X, Super Mario Bros, Keep Tradition Secure, and Targeted Attack: The Game. This will cause the game to become boring. So, the proposed game needs to have an interesting gameplay such as wall slide. Mega Man X , Metal Slug, Keep Tradition Secure, and Targeted Attack: The Game also do not provide tutorials to practice the game control. So, the proposed game will have a tutorial level to let the player familiarize themselves with the game.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This section is used to discuss the methodology for this project. The methodology selected for this project shown in figure 3.0 is Rapid application development (RAD). This is because rapid application development is flexible and adaptable during the development process. This project can be adjusted quickly when facing some problem in the development process. After that, rapid application development also does iterations quickly to reduce the time to develop and speed up the delivery.

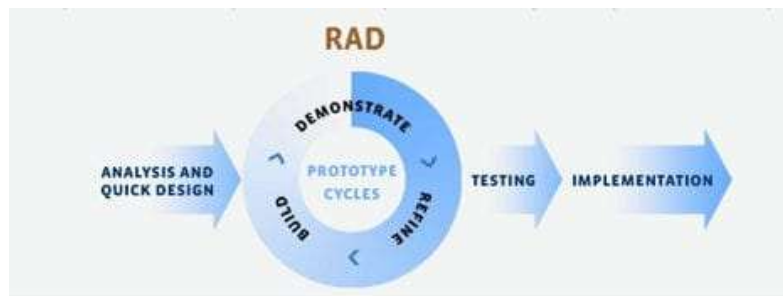


Figure 3.0: The phases of RAD

The process of the game development was developed based on the figure 3.1.

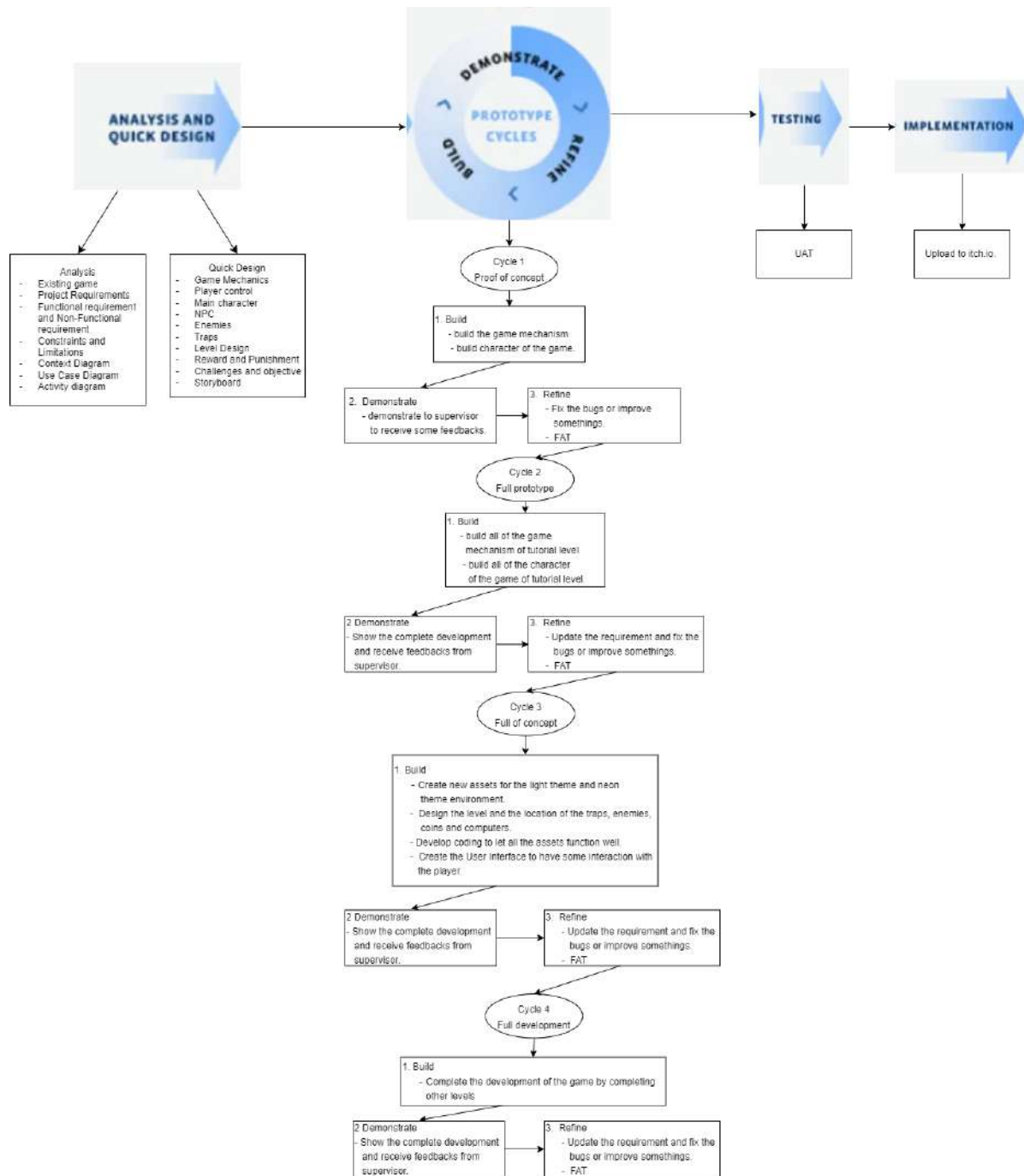


Figure 3.1: The process of the RAD is used for game development.

3.2 Phase 1 - Analysis and Quick Design

This is the first phase of the Rapid application development (RAD) and it is important before starting to develop the proposed game. For the analysis and quick design, there will be

some things to do such as find the requirements of the proposed game, storyboard, context diagram and so on.

3.2.1 Analysis of Existing Games

As a developer, analysis of existing is also important. This is because analysis of the existing game can provide some interesting ideas or game mechanisms to make the game more interesting and provide a help in how to let the player understand the importance of phishing awareness. For example, the wall slide animation shown in Figure 2.4 in section 2.3.1 “Mega Man X” is one of the interesting game mechanisms and can be used in the proposed game to make the game more interesting. (have discuss in chapter 2)

3.2.2 Project Requirement

As a developer, identifying the requirements and carrying out some of the research to develop the 2D platform game about phishing awareness is needed and important. The requirement of the proposed game is to design a clean and simple graphical user interface (GUI) to let the player understand easily. The graphical user interface (GUI) of the proposed game needs to include important information such as score, health bar, and number of coins. The second requirement of the proposed game is the game will be a 2D platform game. The third requirement of the game is there will be a tutorial level to let the player understand and familiar the game rules and the game control. The fourth requirement of the proposed game is there will be some UI pop out about the question of phishing awareness and players need to based on the knowledge learned about the phishing awareness to answer the question correctly. This can increase the knowledge about phishing awareness of the player by answering the question in the game of each level. The fifth requirement of the proposed game is to make the gameplay of the game more interesting such as including the wall slide into the proposed game.

3.2.3 Functional and Non-Functional Requirement

Table 3.0 The functional and non-functional requirements

Functional Requirements	Non-Functional Requirements
<ul style="list-style-type: none"> There are more than 5 game levels. 	<ul style="list-style-type: none"> User interface should be clean, beautiful and simple to let players understand it easily.
<ul style="list-style-type: none"> The life of the character will decrease when colliding with the enemy. 	<ul style="list-style-type: none"> The game does not need the player to key in the personal information and store it.
<ul style="list-style-type: none"> The game will be controlled by using keyboard and mouse. 	<ul style="list-style-type: none"> The reaction and the speed of the character need to be fast enough after the player presses on the button. For example, players need to move forward fast enough within 3 seconds after the player pressed the W button.
<ul style="list-style-type: none"> The game can use the basic button to control the character which is using W,A,S and D buttons to move and jump. 	<ul style="list-style-type: none"> The music of the game needs to match the environment of the level.
<ul style="list-style-type: none"> The player should answer the question about phishing awareness. 	<ul style="list-style-type: none"> The question of the game needs to be asked about phishing awareness.
<ul style="list-style-type: none"> The player needs to collect the coin to install some extension or application. 	

3.2.4 Constraints and Limitations

Table 3.1 Constraints and Limitations

Constraints	Limitations
<ul style="list-style-type: none">• When the character collides with the enemy, the life of character will be deducted.• When the character gets attacked by the phishing attack because they answer the question incorrectly, the money will be deducted.	<ul style="list-style-type: none">• The platform is limited because it is only available on PC.• Only enough money can go to the next level.

3.2.5 Context Diagram

In this section, the game design and the flow of the game are discussed.

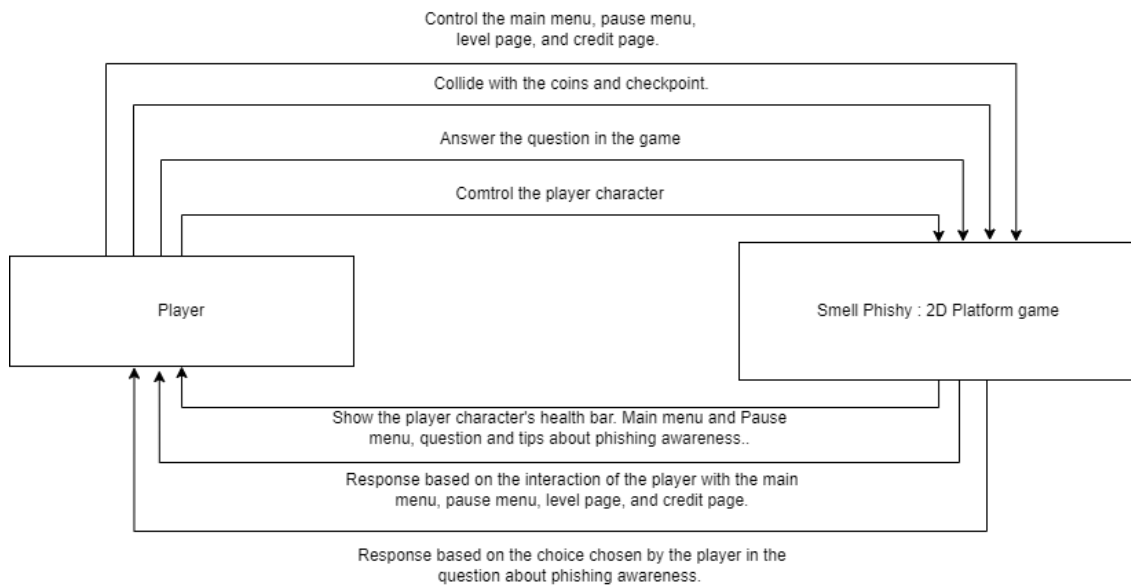


Figure 3.2 : Context Diagram of Smell Phishy, 2D platform game

The figure 3.2 shown context diagram of the proposed game and the context diagram shown 3 major components which are player and the game “Smell Phishy” . Player can interact with the proposed game “Smell Phishy” by using mouse and keyboard. The player can go to the main menu of the game and choose a level, start the game by pressing start game or quit the game. After the player starts the game, the player can press “ESC” to go to the pause menu and the player also can choose to go back to the main menu, restart the level or quit the game. The player can press “W”, “S”, “A”, and “D” to move the character. In the game, there will be some questions about phishing awareness that will pop out to let the player answer and the player can use the mouse to answer the question.

After that, the “Smell Phishy” will show the health bar of the character to let the player know the health left of the player character and the number of the coin collected also will shown in the game UI. The main menu will be shown when the player starts to have an interaction with the game by choosing the level, quit the game and start the game. The game also shows the question after the character collides with some of the NPC and the player needs to answer the question to understand the phishing awareness. Pause menu will pop out after the player presses the “ESC” button and there also some NPC will pop out some tips for the questions that will pop out in the game.

3.2.6 Use Case Diagram

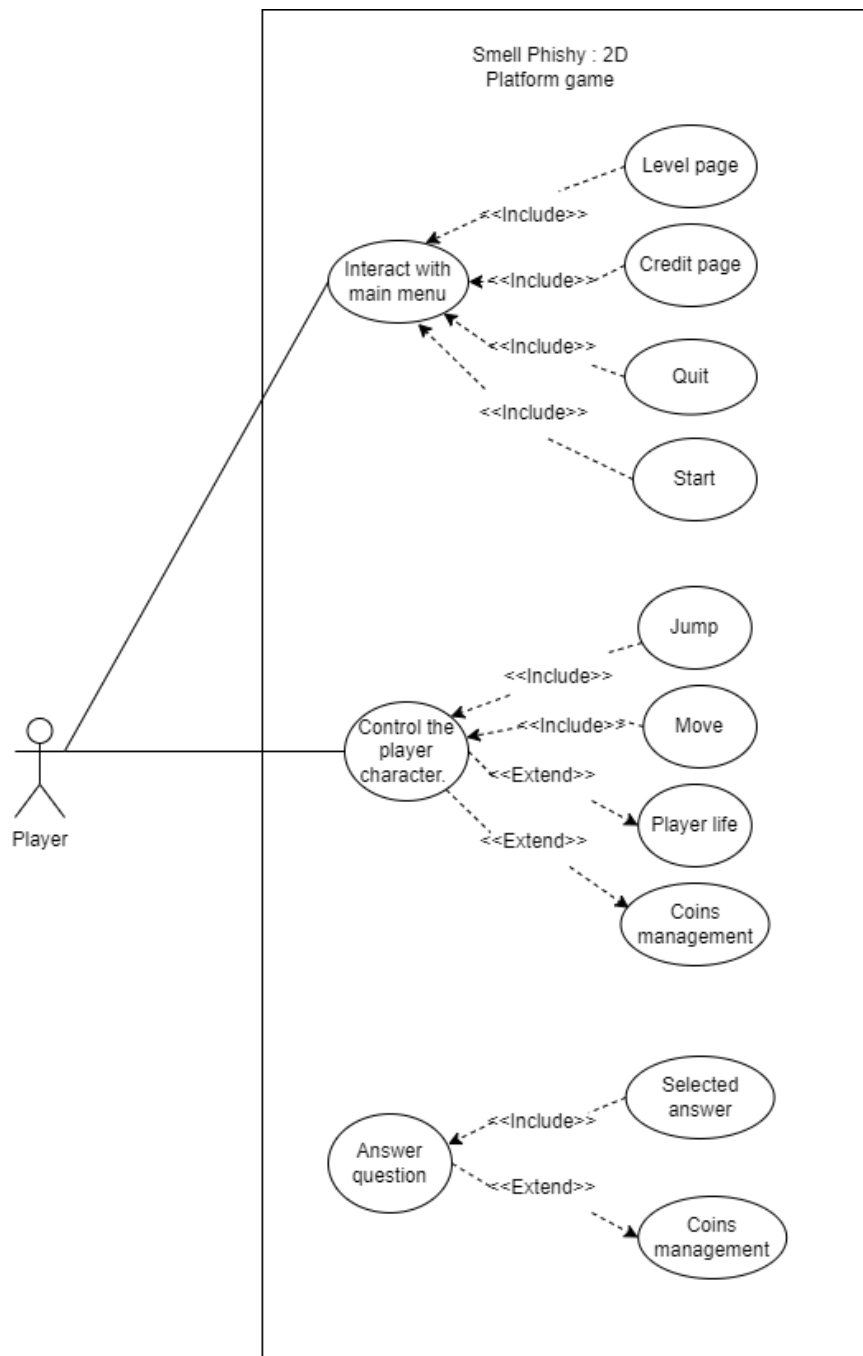


Figure 3.3 : Use Case Diagram of “Smell Phishy”

The figure 3.3 shown the use case diagram of the “Smell Phishy”. In this figure, there is only one player since the game is single player. Players can have some interaction with the game which are start game, quit game, restart game, next level, choose level, pause game, move character, collect coin, answer question, and interact with checkpoint to pass to next level.

3.2.7 Activity Diagram

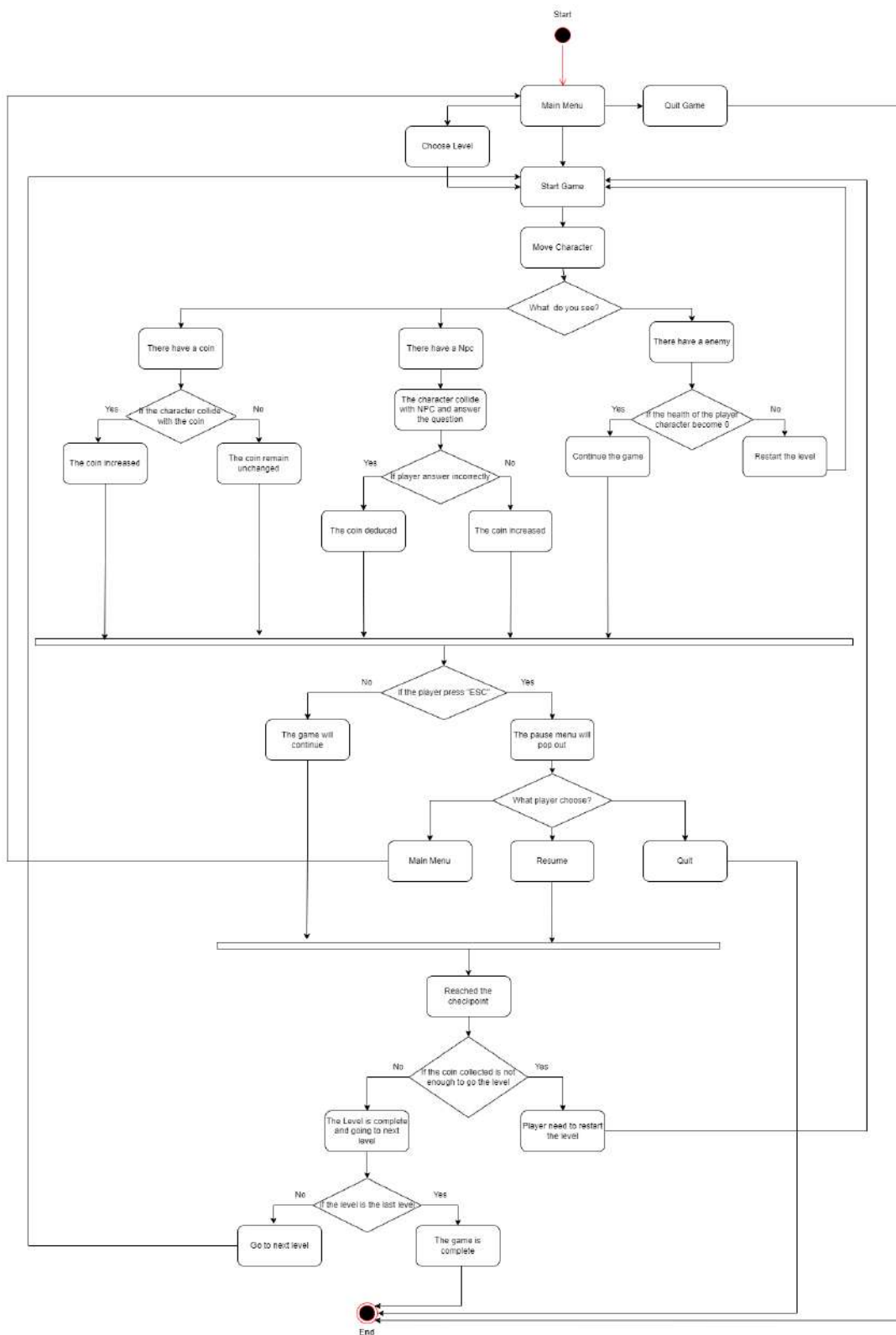


Figure 3.4 : Activity diagram of “Smell Phishy”

In figure 3.4, the main menu will pop out when the game starts. The main menu of the game will show the start game, choose level, and quit game. If the chosen level has been chosen, the game will jump to the level that is selected and start the game. If the start game has been chosen, the game will straight away start from level 1. If the quit game was selected, the game will be exited. After starting the game, the player can move the player character by using the keyboard.

There will be some enemies, coins and some NPCs in the game. If the player character collides with the coin, the coin number that shows in the UI will increase, the coin also will remain unchanged if the player character does not collide with the coin. If the player character collides with the NPC, the NPC will pop out some questions and the player needs to answer it. If the player answers it correctly, the coin will be increased and the coin will be deducted if they answer it incorrectly. There also will be some enemies in the game and player characters need to kill them by jumping or avoid them to prevent the health bar of the character becoming 0. If the player character health bar becomes 0, players need to restart the level of the game.

The pause menu also will pop out if the player presses “ESC” and there are “Resume”, “Main Menu”, and “Quit” to let the player choose on it. If player choose main menu, the game will bring player to main menu of the game, resume will continue the game that the player pause and exit the game if press quit button.

After the player reaches the checkpoint, the player character can pass to the next level if the coin collected is enough. However, the player will need to restart the level if the player did not get enough coins. If the level is the last level, the game has been completed and the game will continue if the level of the game has not finished.

3.2.8 Game Design

3.2.8.1 Game Mechanics

Players need to control the player character to the checkpoint with enough coins collected and pass to the next level. However, there are many enemies that will deduce the health bar of the player character and the player needs to prevent the health bar of the player character from becoming 0. Player also needs to answer the question about phishing awareness and get coins as rewards, the coin also will be deducted if the player answers incorrectly. Players also need to collide with the coins to get enough coins to pass the level.

3.2.8.2 Player Control

The player can use the keyboard to move the player character around in the game. The player can press “W” to move up, “S” to move down, “A” to move left and “D” to move right. The player also can use a mouse to select the answer.

3.2.8.3 Game Character

a. Main Character

The figure 3.5 shows the main character of the game and name “Firefox”. Players need to control the Firefox to move by pressing the “W”, “A”, “S”, and “D” button. Players also need to move the main character to avoid colliding with the enemy to prevent the health bar from becoming 0.

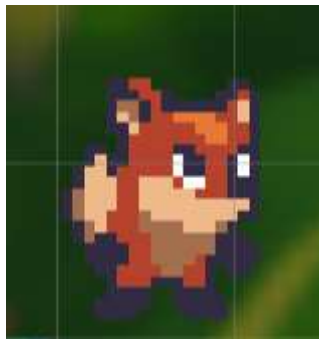


Figure 3.5 : Main Character of the game

b. Non Player Character (NPC)

The figure 3.6 shows the Non Player Character (NPC) in the game and the NPC will give some tips to help players to answer the question about phishing awareness.

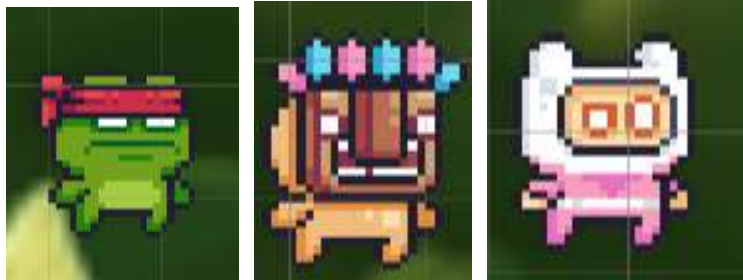


Figure 3.6 : Non Player Character (NPC)

c. Enemy

The figure 3.7 shows the enemy that will be in the game. The pirate will be a scammer that will let the player character lose their life if the player collides with the enemy. The slime will be a virus or malware of the game and players also need to avoid colliding with them.



Figure 3.7 : Enemy

3.2.9 Level Design

Smell Phishy has many different levels and each level has a different question of phishing awareness to let the player answer. The location of the enemy also will be different.

3.2.10 Reward and Punishment

a. Reward

The reward of the game is the coin. Players need to collect the coins to reach the requirement and pass the level. Answering the question correctly in the game also can get some coins.

b. Punishment

The player loses the game when the health bar of the player character becomes 0 and the player needs to restart the game level.


3.2.11 Challenges and Objectives

1. The main objective is that players need to get enough coins to fulfill the requirement and collide with the checkpoint to pass the level.
2. The optional objective of the game is the player can collect all the coins of the game level.
3. The challenge of the game is that players need to avoid or kill the enemy and answer the question correctly to maintain the number of the coins collected that fulfill the requirement of the game.

3.2.12 Storyboard

The following section shows the storyboard for Smell Phishy.

Table 3.2 : Storyboard of main menu

SCENE: MAIN MENU	STORYBOARD 1
	<p>ELEMENTS:</p> <p>Buttons : Start, Level, Credits, and Quit</p> <p>Audio: Background music</p> <p>Sound Effects: Button click sound</p>
<p>SCENE DESCRIPTION</p> <p>The main menu of the game. The title of the game “Smell Phishy” is in the middle of the scene and the scene contain 4 buttons which are “Start”, “Level”, “Credits”, and “Quit”. The Start button can start the game, the level button can let plate choose the game level, credits show some information about the game and quit button will exit the game application.</p>	





INTERACTION	ACTIVITY DESCRIPTION
	<ul style="list-style-type: none"> - When the player clicks on the figure 3.8 which is a Start Button, the main menu will be disabled and the tutorial level of the game will pop out. - The button click SFX will be triggered upon pressing the button.
	<ul style="list-style-type: none"> - When the player clicks Credits Button, the main menu will disabled and the Credits page will pop out. - The button click SFX will be triggered upon pressing the button.
	<ul style="list-style-type: none"> - When the player clicks the Level Button, the main menu will disabled and the Level page will pop out. - The button click SFX will be triggered upon pressing the button.
	<ul style="list-style-type: none"> - When the player clicks Quit Button, the main menu will disabled and the game will be exited. - The button click SFX will be triggered upon pressing the button.

Table 3.3 : Storyboard of Game UI



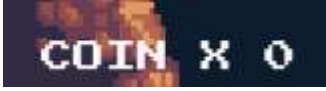
<p>SCENE : GAME UI</p>	<p>STORYBOARD 2</p>
 <p>The screenshot shows a game scene with a character on a platform, a tree, a coin, and a health bar. The health bar is a red bar with a white outline, and the coin is a small gold circle. The text 'COIN X 0' is visible in the top left corner.</p>	<p>ELEMENTS:</p> <p>Audio:</p> <p>Background music</p> <p>Sounds Effects:</p> <p>Jump sound effect, dead sound effect</p>
<p>SCENE DESCRIPTION</p> <p>The Game UI of the game, the player can have a look at the health bar of the player character and the coin collected in the game.</p>	
<p>INTERACTION</p>	<p>ACTIVITY</p>
 <p>A close-up of the health bar, which is a red bar with a white outline and a blue border. The bar is filled with a red, pixelated texture.</p>	<p>When the player character collides with an enemy, the health bar will be deduced and the player will need to restart the game level when the health bar of the player character becomes 0.</p>
 <p>A close-up of the coin counter, which is a black bar with the text 'COIN X 0' in white, pixelated font.</p>	<p>When the player character collides with the coin, the number of the coin will be increased.</p>

Table 3.4 : Storyboard of Credits UI



<p>SCENE : CREDITS UI</p>	<p>STORYBOARD 3</p>
	<p>ELEMENTS :</p> <p>Button :</p> <p>Back</p> <p>Audio :</p> <p>Background music</p> <p>Sound Effects :</p> <p>Button click sound</p>
<p>DESCRIPTION</p> <p>Some information from the developer and a back button can bring the player back to the main menu.</p>	
<p>INTERACTION</p>	<p>ACTIVITY DESCRIPTION</p>
	<p>When the player clicks on the Back button, the Credits UI will disable and the Main Menu will pop out.</p>

Table 3.5 : Storyboard of Level UI




<p>SCENE : LEVEL UI</p>	<p>STORYBOARD 4</p>
 <p>The screenshot shows a futuristic level selection menu titled "Levels". It features a grid of buttons: "TUTORIAL", "LEVEL 2", "LEVEL 3", "LEVEL 4", "LEVEL 5", "LEVEL 6", "LEVEL 7", "LEVEL 8", "LEVEL 9", "LEVEL 10", and a "BACK" button at the bottom center. The background is a blue, glowing globe with circuit-like patterns.</p>	<p>ELEMENTS :</p> <p>Buttons :</p> <p>Tutorial, Level 2, Level 3, Level 4, Level 5, Level 6, Level 7, Level 8, Level 9, Level 10, and Back.</p>
<p>DESCRIPTION</p> <p>Level page UI in the game and there are many buttons to let players choose the level they want to play. The tutorial will be the first level of the game.</p>	
<p>INTERACTION</p>	<p>ACTIVITY DESCRIPTION</p>
 <p>A close-up of the "LEVEL 2" button, which is a green rounded rectangle with a glowing border and the text "LEVEL 2" in white.</p>	<p>When the player clicks on the Level 2 button, the level UI disables and goes to the level 1 of the game.</p>
 <p>A close-up of the "BACK" button, which is a green rounded rectangle with a glowing border and the text "BACK" in white.</p>	<p>When the player clicks on the Back button, the Credits UI will disable and the Main Menu will pop out.</p>

Table 3.6 : Storyboard of Game Complete UI

<p>SCENE : GAME COMPLETE UI</p>	<p>STORYBOARD 5</p>
	<p>ELEMENTS :</p> <p>Buttons :</p> <p>Next Level, Main Menu, Quit</p>
<p>DESCRIPTION</p> <p>A UI of the game complete panel will pop out after the player completes all of the level of the game.</p>	
<p>INTERACTION</p>	<p>ACTIVITY DESCRIPTION</p>
	<p>When the player clicks on the Next level button s, the Game Complete UI disables and goes to the next level of the game.</p>
	<p>When the player clicks on the Main Menu button, the Game Complete UI disables and goes to the main menu of the game.</p>
	<p>When the player clicks on the Quit button, the Game Complete UI disables and exit the game.</p>

Table 3.7 : Storyboard of Pause menu UI

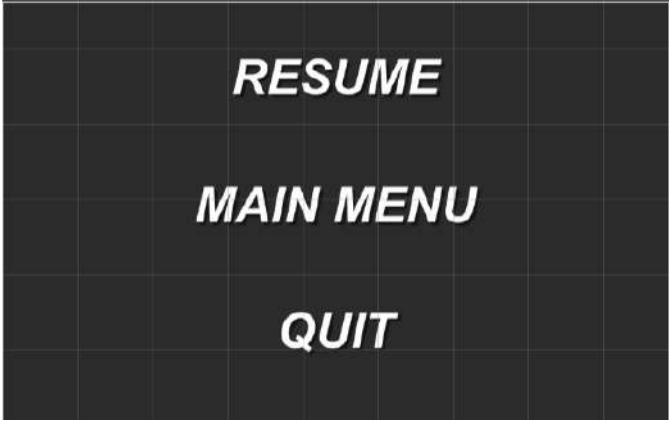



SCENE : PAUSE MENU UI	STORYBOARD 6
	<p>ELEMENTS :</p> <p>Buttons :</p> <p>Resume, Main menu, Quit</p>
<p>DESCRIPTION</p> <p>The pause menu UI will pop out when the player presses “ESC” in the game.</p>	
INTEARCTION	ACTIVITY DESCRIPTION
	<p>When the player presses on the resume button, the game will continue where the player pauses.</p>
	<p>When the player presses on the main menu button, the game will bring the player to the main menu of the game.</p>
	<p>When the player presses on the quit button, the game will exit.</p>

Table 3.8 : Storyboard of health bar of player character


<p>SCENE : HEALTH BAR OF PLAYER CHARACTER</p>	<p>STORYBOARD 7</p>
	<p>ASSETS REQUIRED:</p> <p>Audio : Chill, relax and attractive music</p> <p>Sound : hurt sound effects</p>
<p>DESCRIPTION Health bar of the player character will be deduced when the player character collides with the enemy.</p>	

Table 3.9 : Storyboard of ladder


<p>SCENE : LADDER</p>	<p>STORYBOARD 8</p>
	<p>ASSETS REQUIRED:</p> <p>Audio : Chill, relax and attractive music</p> <p>Animation : Climb</p>
<p>DESCRIPTION Player character can climb on the ladder by pressing the “W” button.</p>	

Table 3.10 : Storyboard of player character with the coin



<p>SCENE : PLAYER CHARACTER WITH COIN</p>	<p>STORYBOARD 9</p>
	<p>ASSETS REQUIRED:</p> <p>Audio :</p> <p>Chill, relax and attractive music</p> <p>Sounds :</p> <p>Coin collect sound effect</p> <p>Animation :</p> <p>coin rotate animation</p>
<p>DESCRIPTION</p> <p>The coin will disappear and the coin sound will trigger when the player character collides with the coin.</p>	

Table 3.11 : Storyboard of question about phishing

<p>SCENE : QUESTION ABOUT PHISHING</p>	<p>STORYBOARD 11</p>
	<p>ASSETS REQUIRED:</p> <p>Audio :</p> <p>Chill, relax and attractive music</p> <p>Button:</p> <p>Phishing and Legitimate button</p>
<p>DESCRIPTION</p> <p>The question will pop out when a player collides with some things or NPC.</p>	

3.3 Phase 2 - Prototype Cycles

Prototype cycles is the second phase of Rapid Application development. In this phase, there are three steps and three of these steps can be loop until the proposed project reaches the requirement. There are 4 prototype cycles, 2 cycles are used in the development of the prototype and 2 cycles are used in the full development process of the game.

3.3.1 Build Prototype

This process is important and it is the starting point for building a prototype of the game. The prototype will contain some basic functions but there are also many bugs in the proposed game and it was not a complete game. The prototype also will insert all the assets that need to be used in the proposed game.

3.3.2 Demonstrate

a. Demonstrate prototype

The prototype will be demonstrated to the project supervisor by having a meeting. After that, the second people who are going to have a look are friends. All of them will give some feedback after having a try with the prototype built .

b. Gather feedbacks

The feedback is needed for improving the project. After receiving all the feedback from project supervisor and friends, the proposed game will be carried on to the next step which is refinement. The feedback can help to improve or fix the bug that was founded by friends or project supervisor.

3.3.3 Refine

a. Update the requirement of proposed game

The requirement of the proposed game will be update, add or change based on the feedback that gathers from the phase demonstration.

b. Fix the bug

The bug of the proposed game also will be fixed after receiving some feedback from the project supervisor and friends. After fixing all the bugs of the proposed game, the prototype cycles will be repeated until the proposed game reaches all the requirements and all the bugs of the game are fixed.

Requirement form

Table 3.12: Requirement form

No	Requirement	Action
1.	Make the User interface of the game, Main Menu, and Quiz panel more tidy, clean and easy to let users understand.	<ol style="list-style-type: none">1. Put some image as the decoration on the health bar, requirement and coin to make the UI clean and tidy.2. Make the User Interface Main menu, and Quiz panel more tidy and stick to the topic.
2.	Think new theme for the other environment	<ol style="list-style-type: none">1. There are 3 themes used in the game application which are normal, light and neon.
3.	Change the environment to stick with the topic which is Phishing awareness.	<ol style="list-style-type: none">1. Change the prototype background and assets to stick with the topic.

Functional acceptance test

Functional acceptance test is used to uncover structural problems, hidden errors, and problems with specific components. The code of the proposed game needs to be updated or changed in a refined process after getting feedback from the demonstration process.

Table 3.13: Template of Functional acceptance testing

Test Case ID	Test Case	Test Data	Test step	Expected Result	Actual Result	Pass /Fail	Note/ Action
TC 01	Check the response when player press on the movement button	press “W”, “S”, “A”, and “D” button	i) launch the game ii) press the movement button	The player will move on the platform	The player just moving left and right, but the player did not flip when moving opposite direction	Fail	I) Check the PlayerMovement script II) Add “ <code>sprite.flipX = false;</code> ” into the PlayerMovement script.

3.4 Phase 3 - Testing

Testing process is very important and necessary to make sure the function of the project can be functioning and this phase will need to carry out a User Acceptance Testing (UAT). This UAT was carried out where users needed to test all the available features in the game application and check the functionality of the game and another set of UAT was used to test the usability and effectiveness of the game application. There will be 20 responses in the testing section. The UAT Testing form can refer to Appendix 1.

3.5 Phase 4 - Implementation

The implementation phase is the last phase of the development of the proposed game. The Smell Phishy was developed by using Unity and the assets were found in the assets store.

3.6 Conclusion

As a conclusion, the methodology that was used to develop the Smell Phishy is Rapid Application Development (RAD). This methodology is suitable to the development of the Smell Phishy because Rapid Application Development (RAD) is easy, simple and flexible to make changes in the half way of the development.

For the first phase of the Smell Phishy, the name of the phase is analysis and quick design. This phase is important before starting the development of the proposed game. In this phase, the analysis of existing games is needed and having a look at the existing game can provide some ideas and some interesting game mechanisms can make the game more interesting. Analyzing existing games also can provide some weakness and improve the weakness when developing the proposed game "Smell Phishy ". After that, the requirement also needed to develop a proposed game. Requirement of a game is very important because it is a starting point and an idea to make a game.

The second phase of Smell Phishy is Prototype cycles. In prototype cycles, there have three things needed to do which are build, demonstrate and refine. In building the prototype of the game, the prototype needs to follow the requirements that have been set in phase one which is analysis and quick design. After building the prototype, the prototype will be demonstrated in the demonstration phase. In this phase, the feedback will be gathered and used to update the requirements of the proposed game. Refine phase will be carried out after the demonstration phase and all the bugs or hidden errors will be tried to fix.

For the testing phase, there will be several tests to test the game to find the hidden error and fix them. This phase will be more detailed compared to the refine phase and there will be a white box test and black box test to test the proposed game. After that, Implementation will be carried out and the development of the game was developed by using unity.

After all the phases, the development of the Smell Phishy was finished and the game will update to fix the bug that will appear in the future.

CHAPTER 4

DEVELOPMENT PROGRESS

4.1 Introduction

Chapter 4 will discuss the third, fourth and fifth times of the recycles of the prototype cycles, testing of “Smell Phishy”, and implementation. Unity, Microsoft Visual Studio 2019, Adobe Photoshop and Kawning are used to develop the game application. Testing is the process to test the game “Smell Phishy” to make sure the game can function well before the next process. Implementation is the last process to debug the game and maintenance tasks before publishing to the public.

4.2 Prototype cycles Process

The prototype cycles have been recycled 2 times in chapter 3 to make a full prototype. In chapter 4, the prototype cycles also recycle 2 times to make a complete 2D platform game. In the prototype cycle, there are 3 steps that need to be processed to complete the game. The table 4.2.1 showed the prototype of the cycles process.

Table 4.2.1: Prototype cycles process

First cycles	Second cycles
<p>Build</p> <ol style="list-style-type: none">1. Continue to develop for the light and neon theme environment.2. Create new assets for the light theme and neon theme environment.3. Design the level and the location of the traps, enemies, coins and computers.4. Develop coding to let all the assets function well.5. Create the User Interface to have	<ol style="list-style-type: none">1. Complete the development of the game by completing other levels of the game.

some interaction with the player.	
<p>Demonstrate</p> <ol style="list-style-type: none"> 1. Show the complete development of the process about light and neon theme environments. . 2. Receive feedback from the supervisor. 	<p>Demonstrate</p> <ol style="list-style-type: none"> 1. Show the complete development of the process. 2. Receive feedback from the supervisor.
<p>Refine</p> <ol style="list-style-type: none"> 1. Update the requirements of the game. <ul style="list-style-type: none"> - User interface of main page and level page not suitable - Question of the game not suitable - Level page cannot function well. <p>Fix bug</p> <ol style="list-style-type: none"> 1. Fix the bug of the level page. 2. Fix the problem of User interface. 3. Redo the question of the game. 	<p>Refine</p> <ol style="list-style-type: none"> 1. Update the requirement of the game <ul style="list-style-type: none"> - Some of the assets did not function well. <p>Fix bug</p> <ol style="list-style-type: none"> 1. Find the problem of the code and fix it by changing the code. 2. Fix the loading problem of the User interface.

4.2.1 Build

This build process is the first step of the prototype cycle and this step is going to start to build a complete game. The development tools that were used to build the game, design the graphic and multimedia contents, develop the scene of the game, and coding that created for the game was written under the build process.

For the first cycle of the prototype cycle, the build process was used to build up all the assets and located into the game application to make a full complete game. However, for the second cycle of the build process, the game was put into the build process again after getting

some opinion from the supervisor and friends. The second build process was going to insert some new assets or new environments such as other themes of the game application to the new environment.

A. Development Tools

There are some of the development tools used to develop this game application and shown in table 4.2.2.

Table 4.2.2 : Tools used for the development of game application

No	Tool	Purpose
1	Unity	To create the game platform, game content, character, enemy and gameplay
2	Microsoft Visual Studio 2019	For scripting
3	Adobe Photoshop	To create and edit image
4	Kawing	To edit and cut audio

B. Designing graphic and multimedia Contents

The graphic and multimedia contents are very important to make an attractive and beautiful game. The graphic and multimedia contents that included are image, text, audio and animation. At this stage, there will be some assets that will change to improve the user interface and the gameplay of the game.

Image is the most important asset for a game because it represents the theme of the game and the graphic of the game. For example, if the theme of the background image is island and the theme of the game is city, this situation will make the game become weird because the background image is not the same as the theme of the game. Some of the background image, enemy image, and button image are edited using Adobe Photoshop and some of it is obtained from the internet. Figure 4.2.1.1 and Figure 4.2.1.2 shows the sample of the editing image in Adobe Photoshop. The animation of the player character, enemy and

trap are obtained from the internet and edited in Adobe photoshop to get a higher quality image for the game. Figure 4.2.1.3 shows the animation that was edited in Adobe Photoshop. Figure 4.2.1.4 also shows the 2D sprites used in developing the game content.

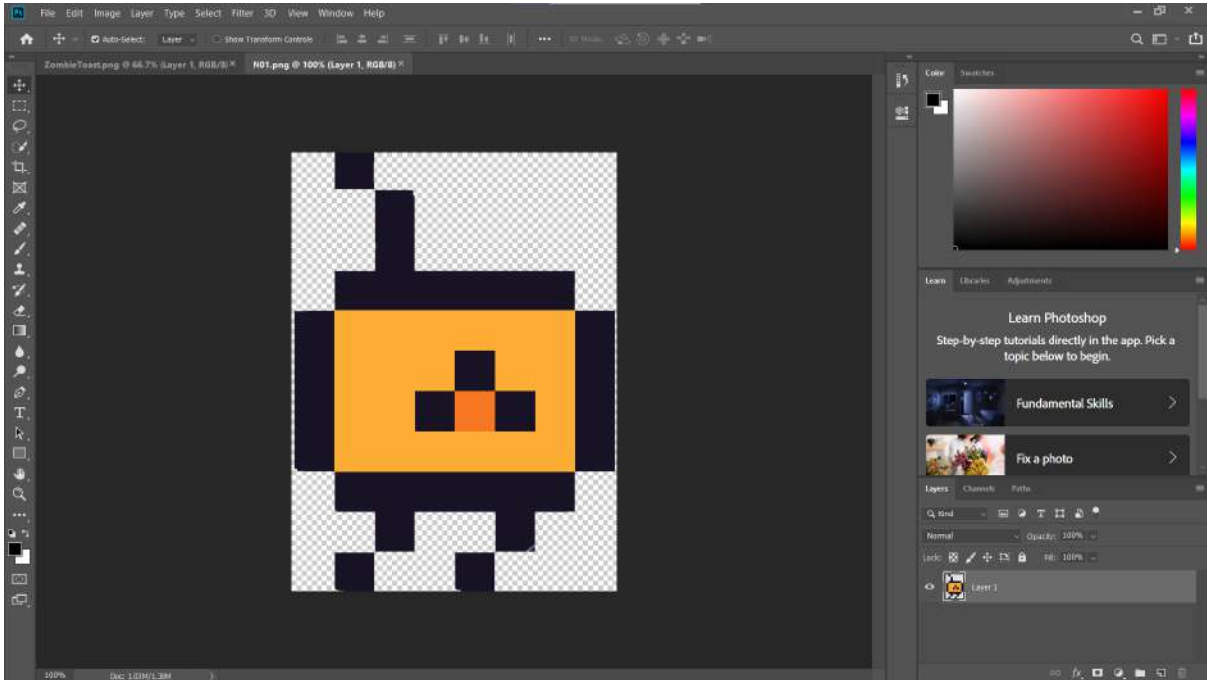


Figure 4.2.1.1 : Example of editing NPC in Adobe Photoshop

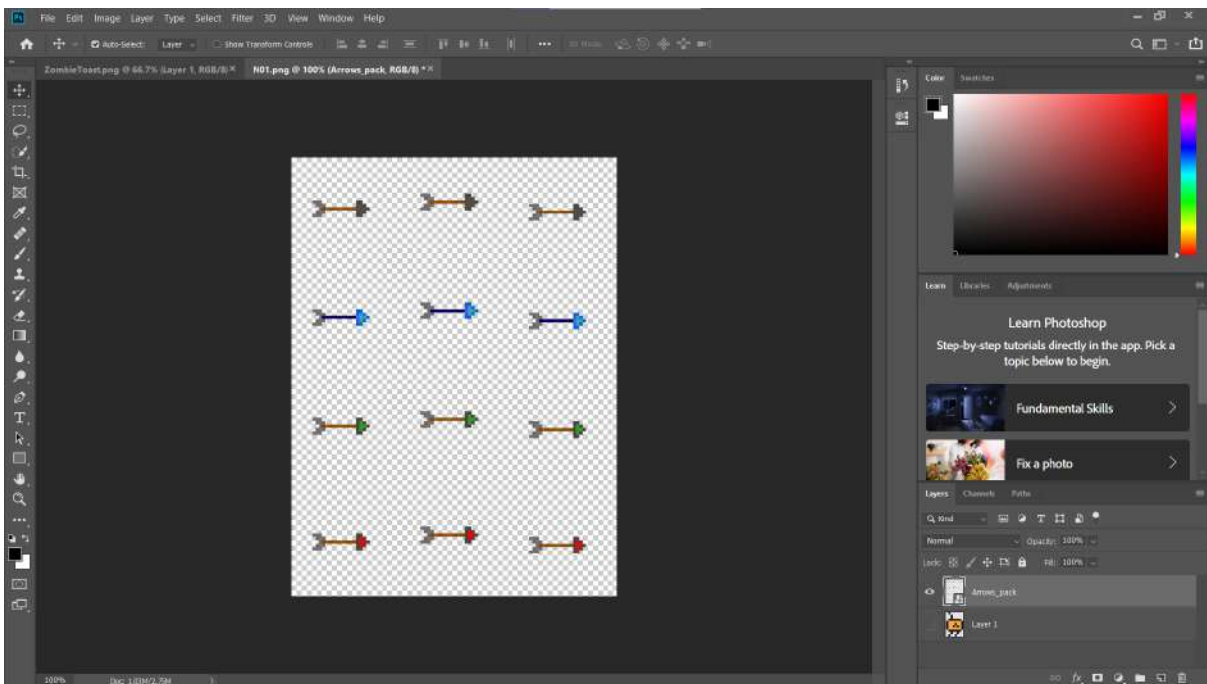


Figure 4.2.1.2: Example of editing image arrow as trap in Adobe Photoshop

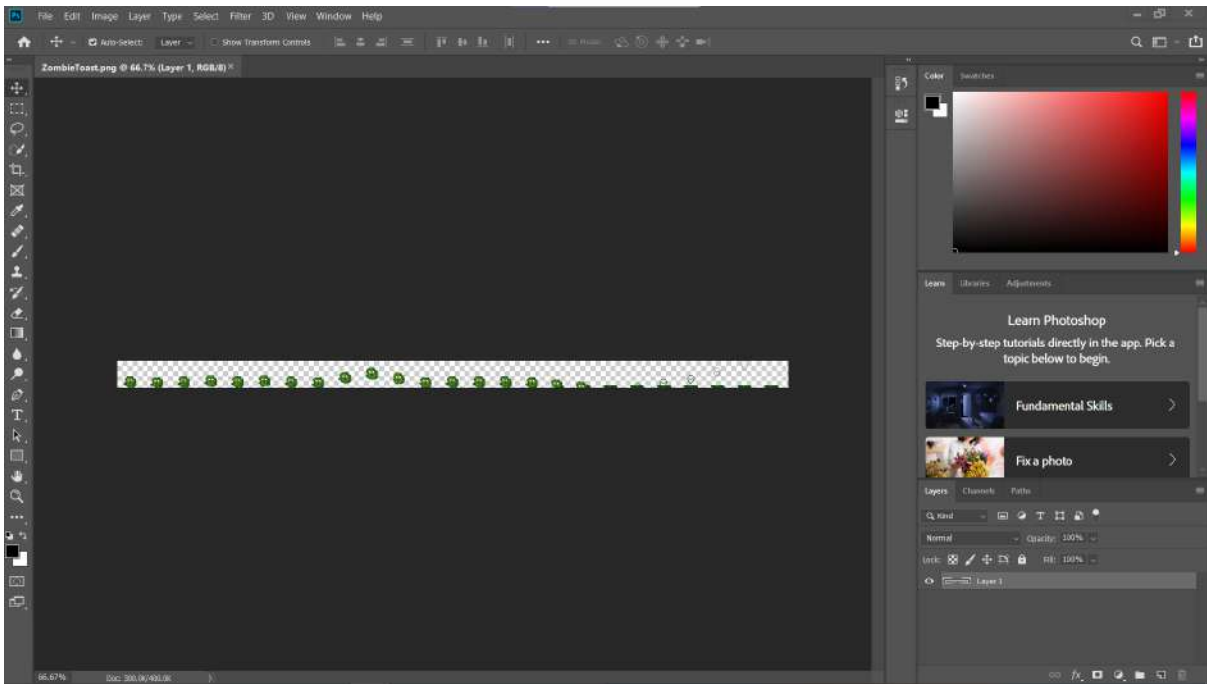


Figure 4.2.1.3: Animation of enemy editing in Adobe photoshop

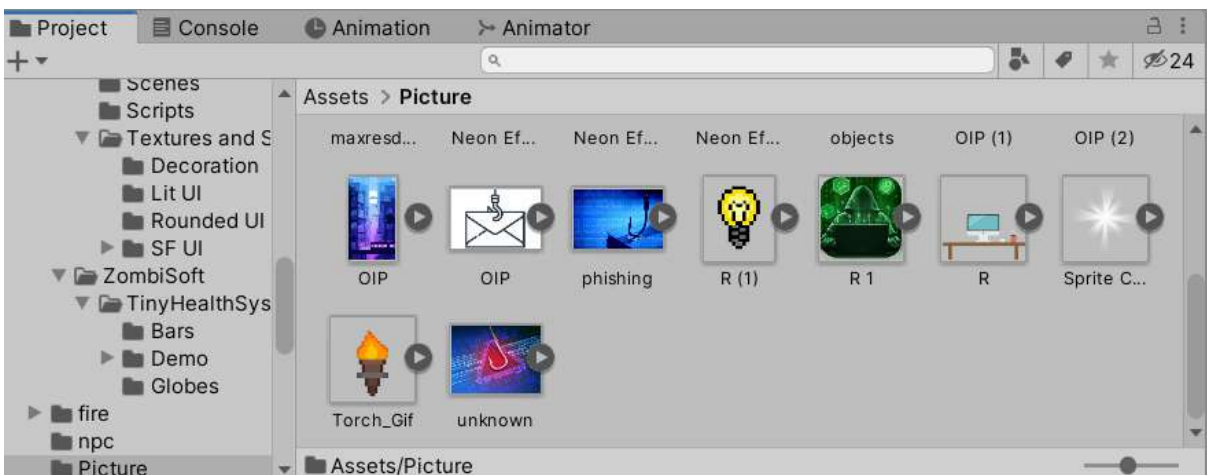


Figure 4.2.1.4 : Some example of 2D sprites that used in the game application

After that, text is also very important to transfer some information to the player. The text will act as an interaction between the player and game. The text that is used in the game is PressStart2P-Regular. Figure 4.2.1.5 shows the text that is used in the UI of the game.

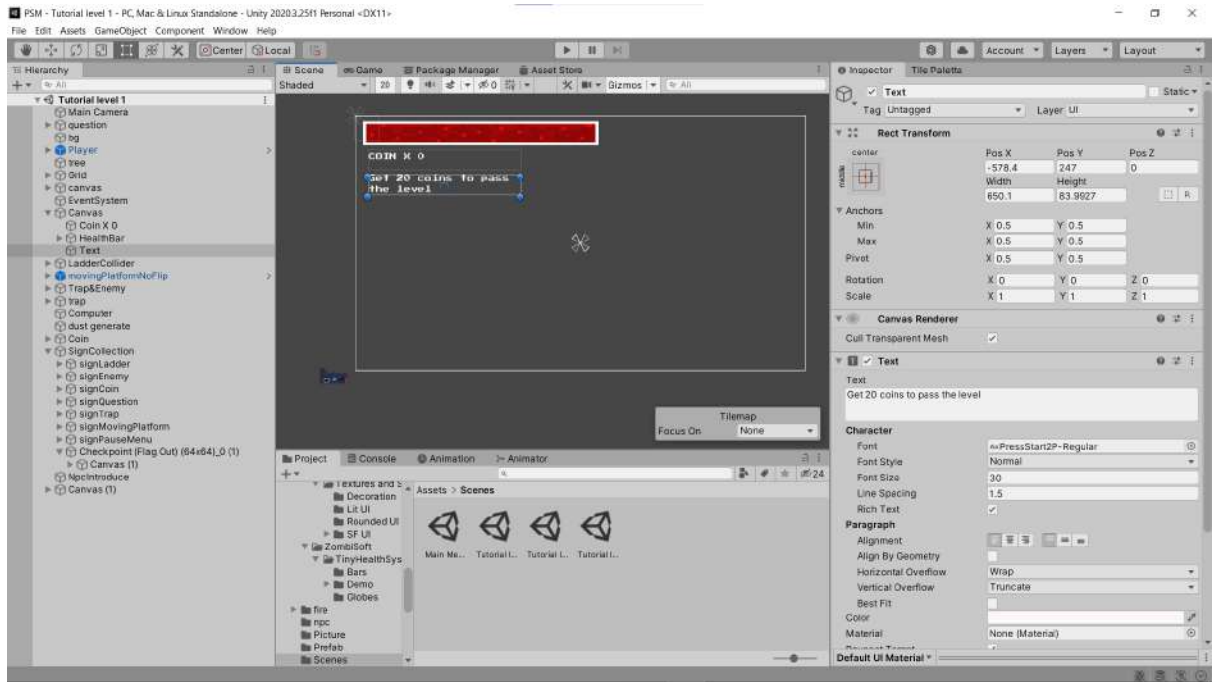


Figure 4.2.1.5 : Text “PressStart2P-Regular” used as the game UI

Besides that, some of the audio is from the internet and some of it is edited by using Kawing which is an online application. The sound effect is applied when the player character dies, jumps, and some background music is also applied in a different environment. Figure 4.2.1.6 shows the audio that was edited in the kawing.

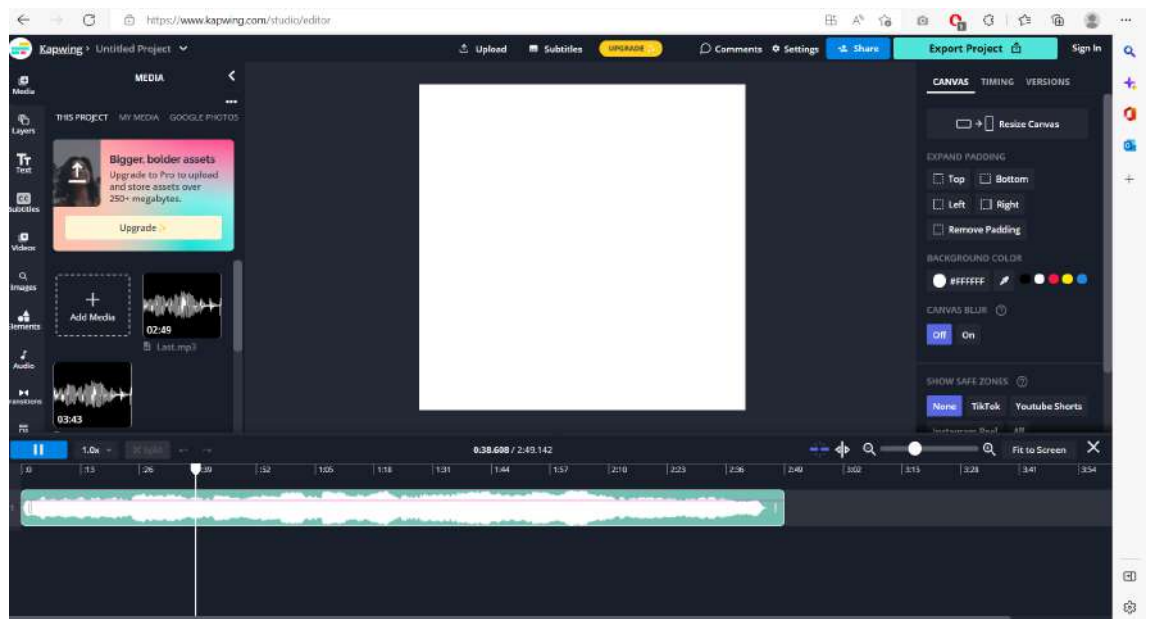


Figure 4.2.1.6 : Audio editing in Kawing

C. Level Design

In this section which is level design, the theme and the design of the level was discussed. There are 6 levels for the game and there are also 3 themes used in the game which are normal, light and neon.

Table 4.2.2 : Level design

Levels	Theme
1	Normal
2	Normal
3	Light
4	Light
5	Neon
6	Neon

The table 4.2.2 shows the level design of the game. There are 6 levels of the game application and the level design of the 6 levels are different. There are 3 themes for each 2 levels which are normal, light and neon. For the first 2 levels the tutorial has the same environment and same enemies, traps and moving platform. However, level 2 will be longer compared to the tutorial and difficult. Level 3 and 4 having the new environment and the theme using is light. New environment also has new enemies, traps, and moving platforms. However, the difference between level 3 and 4 is that level 4 is deeper than level 3. For level 5 and 6, there is a new theme which is neon and there are new environments, new traps and new moving platforms. The level 6 will be higher than level 5.

D. Development of the Game Application

This section is used to describe the development progress of the game application of "Smell Phishy". All of the scene, code, and interface that in the game application are described in this section.

E. User Interface

A. Main Menu page

Figure 4.2.1.7 shows the main menu of the application. The main menu has four buttons, which are “ Start” , “Level”, “ Credits” and “ Quit”. All of the buttons have their own function. The green picture in the main menu is the logo of the game and the “SMELL PHISHY” is the name of the game application. For the “Start” button, it navigates the player to start the game application and the game starts with the tutorial to guide the player. The “Credits” button brings the player to the credits page and there are some details about the developer in the credit page. “Level” button brings the player to the level page and there are 6 levels to let the player choose the level.



Figure 4.2.1.7 : Main Menu Scene

Figure 4.2.1.8 shows the inspector of the Start button in the main menu page. The source image in the image section chose the UI2_3 as the image of the button and it was also used in the Credits, Level, and Quit button. The function of the button was set in the OnClick() by dragging the object in the hierarchy that was inserting the code name “Main Menu” into it. After clicking on the button, the audio was played because of the function of AudioSource.Play that was set in OnClick() and jumped to the tutorial level because of the

function of MainMenu.Tutorial. There is also an animation of the button that is set in the animator.

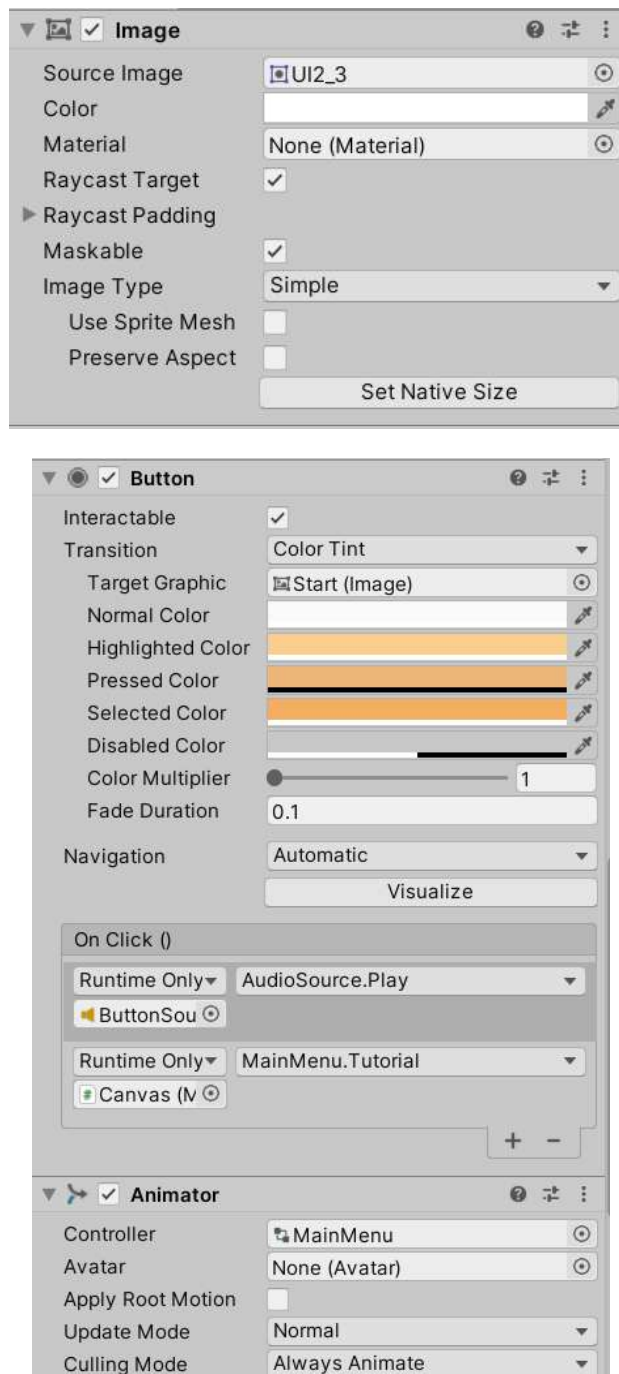


Figure 4.2.1.8 : Inspector of Start button

Figure 4.2.1.9 shows the different function compared to the start button. The game will bring the player to the level page and close the main menu page because of the function that is set on the On Click () which is `GameObject.SetActive` to activate the level page and

deactivate the main menu page. The audio was played when the player clicked on the button after the AudioSource.Play was set.

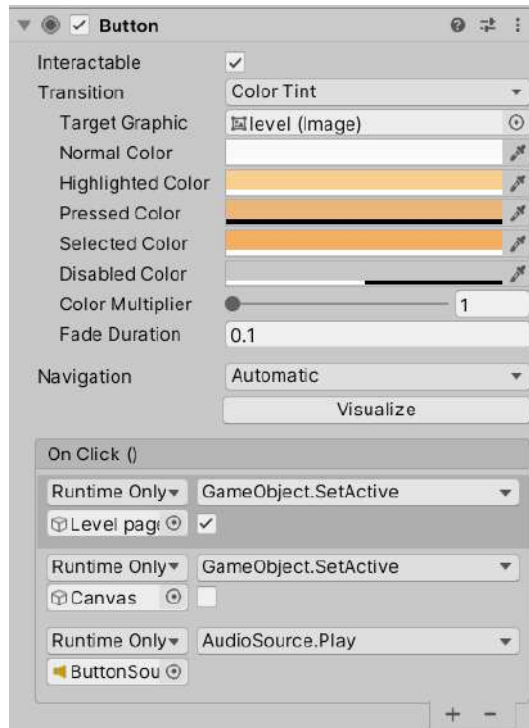


Figure 4.2.1.9: Inspector of level button

Figure 4.2.1.10 shows the different functions compared to the level and start button. The first On Click() can bring the player to the credit page and the second On Click() used to disable the main menu page. The AudioSource.Play was used to play audio after the player clicked on the button.

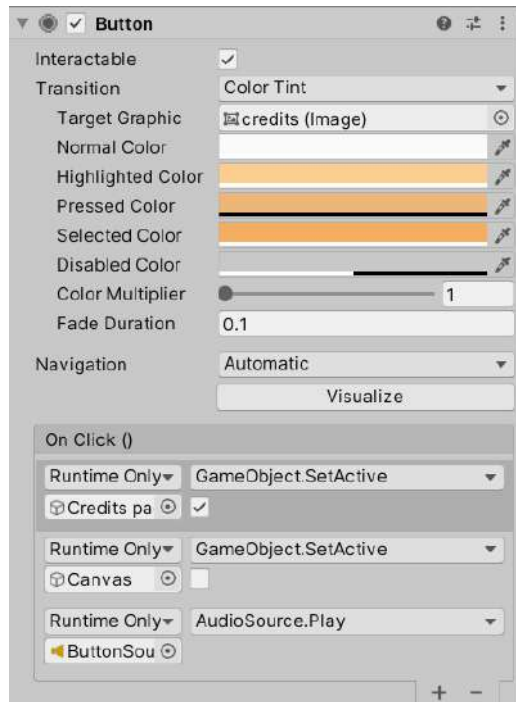


Figure 4.2.1.10: Inspector of Credit button

Figure 4.2.1.11 shows the inspector of the quit button. Drag the Credits page into the first On Click() and activate the by selecting the GameObject.SetActive and tick it. After that, the second On Click() was by dragging the main menu and deactivating it by selecting the GameObject.SetActive.

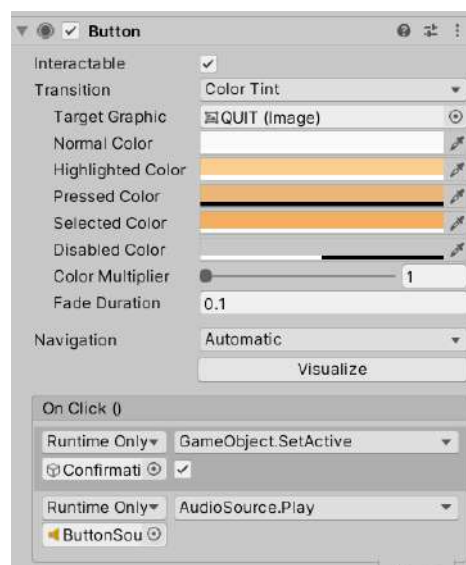


Figure 4.2.1.11 : Inspector of Quit button

The quit button used the function that was set in the exit() of the script in the figure 4.2.1.12. The Tutorial(), Level2(), Level3(), Level4(), Level5(), and Level6() were used to bring the player to each of the levels. MainMenuPage() was used to bring the player to the main menu and Sound() can be used to trigger button sound when the player clicks the button. The full script was shown in appendix 1.

```

20     public void exit()
21     {
22
23         Application.Quit();
24         Debug.Log("Quit");
25     }
26     public void Tutorial()
27     {
28
29         SceneManager.LoadScene("Tutorial level 1");
30         FindObjectOfType<PlayerLife>().resetScore();
31     }
32
33     public void Level2()
34     {
35
36         SceneManager.LoadScene("Tutorial level 2");
37         FindObjectOfType<PlayerLife>().resetScore();
38     }
39
40     public void Level3()
41     {
42
43         SceneManager.LoadScene("Tutorial level 3");
44         FindObjectOfType<PlayerLife>().resetScore();
45     }
46
47     public void Level4()
48     {
49
50         SceneManager.LoadScene("Tutorial level 4");
51         FindObjectOfType<PlayerLife>().resetScore();
52     }
53
54     public void Level5()
55     {
56
57         SceneManager.LoadScene("Tutorial level 5");
58         FindObjectOfType<PlayerLife>().resetScore();
59     }
60
61     public void Level6()
62     {
63
64         SceneManager.LoadScene("Tutorial level 6");
65         FindObjectOfType<PlayerLife>().resetScore();
66     }
67
68     public void MainMenuPage()
69     {
70
71         SceneManager.LoadScene("Main Menu");
72         FindObjectOfType<PlayerLife>().resetScore();
73     }

```

...roject_unity\PSM\Assets\Script\User Interface\MainMenu.cs 2

Figure 4.2.1.12: Script of Main menu

B. Confirmation message

Figure 4.2.1.13 shows the confirmation message after clicking on the Quit button. Players can press the “Yes” button to quit the game or click on the “No” button to return to the main menu page.

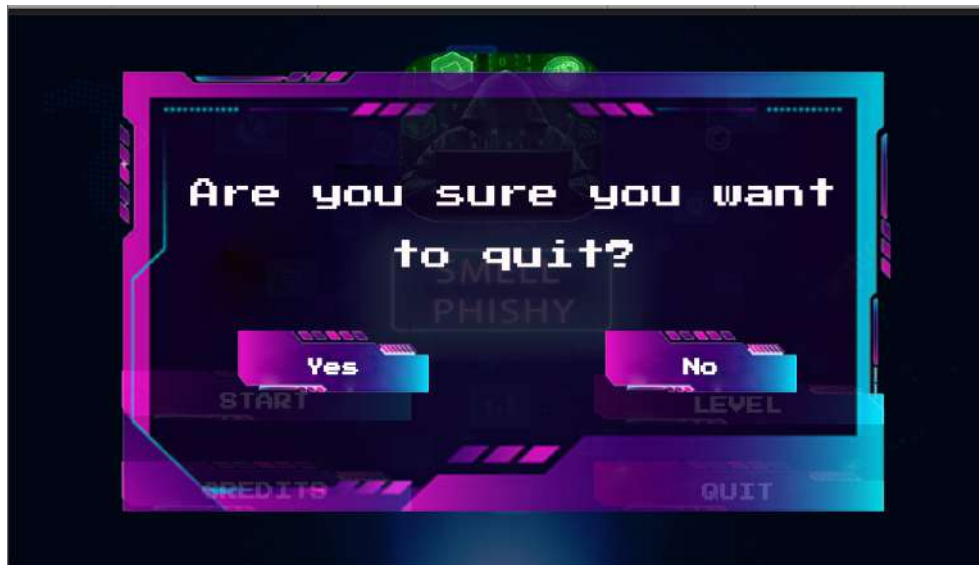


Figure 4.2.1.13 : Confirmation message.

Figure 4.2.14 shows the inspector of the “Yes” button in the confirmation page. After clicking on the Yes button, the game will exit because of the function of the first OnClick(). The first OnClick() was set by dragging the Canvas (Main Menu) that included the Main Menu code. After clicking the button, there is a sound coming out because of the function of OnClick() of AudioSource.Play.

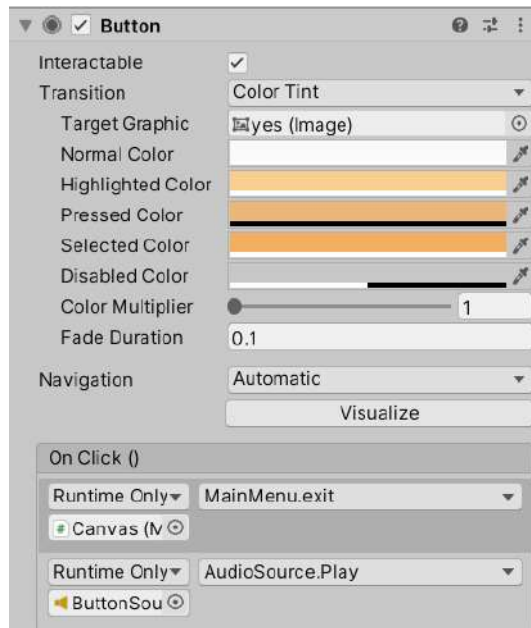


Figure 4.2.1.14: Inspector of “Yes” button in confirmation page

Figure 4.2.1.15 shows the inspector of the “No” button. The OnClick() are used to deactivate the confirmation page and activate the Main Menu page and there a sound comes out after clicking on the button.

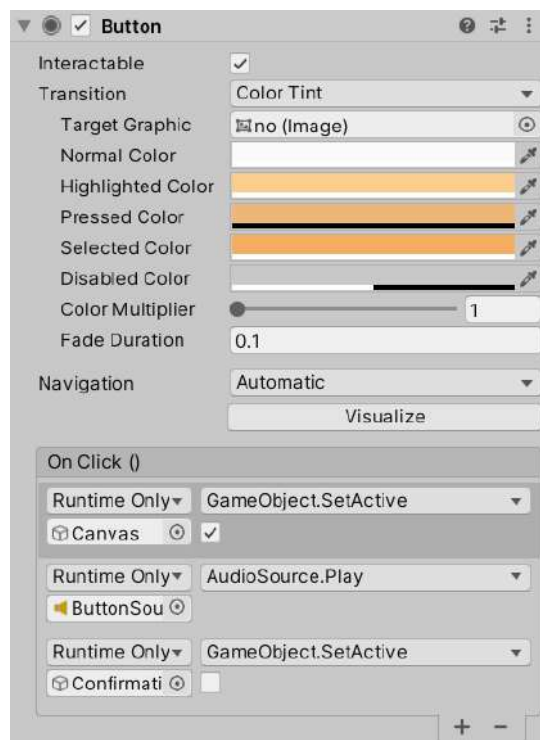


Figure 4.2.1.15: Inspector of “NO” button.

C. Credits page

Figure 4.2.1.16 shows the credits page after pressing the “Credits” button on the main page. The page shows some of the details of the developers and the “Back” button brings the player back to the main menu page.



Figure 4.2.1.16: Credits page.

Figure 4.2.1.17 shows the inspector of the back button. The function of the button was set in OnClick(), deactivated the credit page and activated the main menu page. The sound of the button also pops out.

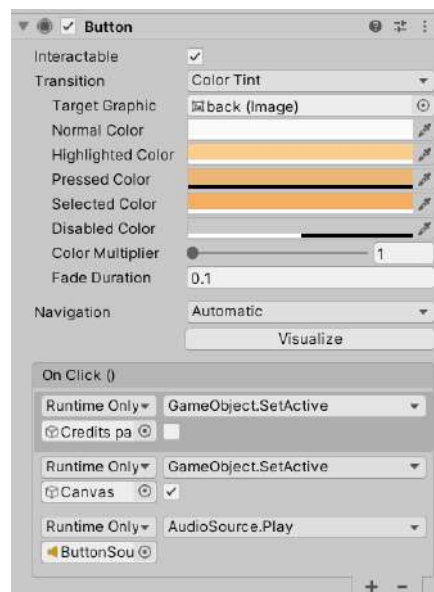


Figure 4.2.1.17: Inspector of back button.

D. Level Page

Figure 4.2.1.18 shows the levels page after pressing the Level button on the main page. All of the Level buttons are built up by inserting the button function in the inspector. So, players can interact with the Level button. For the scroll view of the game, mask and horizontal layout group insert into the inspector. The level page can be scrolled from left to right by dragging using the mouse to see more levels in the game application. The level is locked by using the script name “LevelSelection” and inserted into the inspector. The Reset button will lock the level again except tutorial level to let the player replay the game application. The “<” button brings the player back to the main menu page if pressing on it.

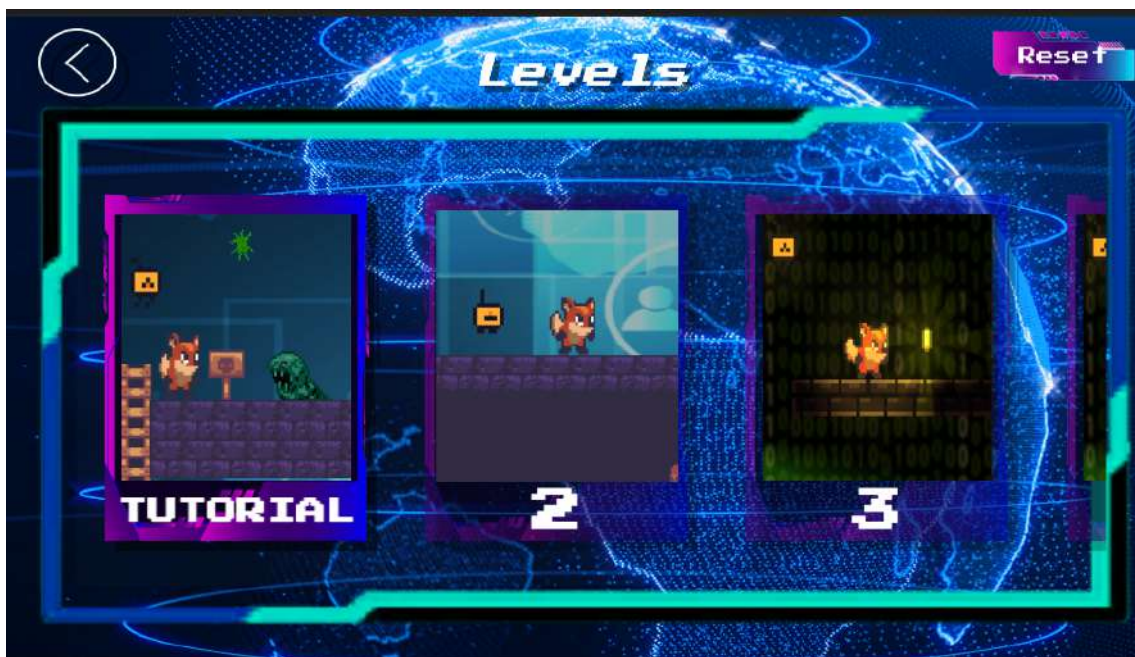


Figure 4.2.1.18: Level page

Figure 4.2.1.19 shows the inspector of the level. It can let the level page have a horizontal scroll view. Dragging the level select to the viewport to let the level select fix in the scroll view.

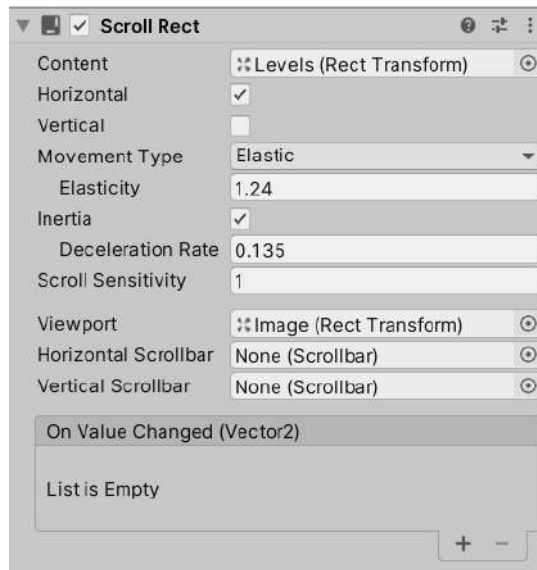


Figure 4.2.1.19: Inspector of the Level.

The figure 4.2.1.20 shows the inspector of masks that insert in the level page. It used to let the level selection fix into the scroll view.



Figure 4.2.1.20: Mask Inspector

The reset button was used to reset the level if the player passed the level and wanted to replay all the level. Figure 4.2.1.21 shows the inspector of the reset button and all of the function was set in On Click(). The first OnClick() set the function to reset the level that was passed and leave the tutorial level. The audio also pops out after the player click the button.

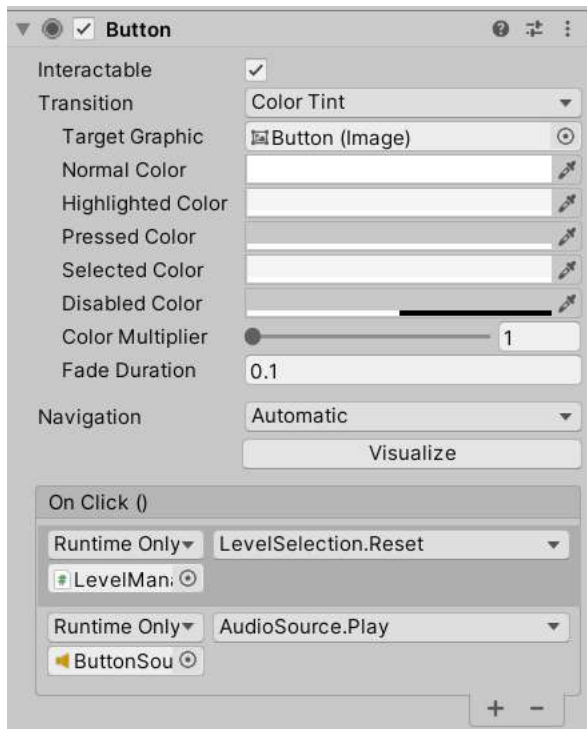


Figure 4.2.1.21: Inspector of the Reset button

The figure 4.2.1.22 shows the inspector of the back button. The OnClick() was used to set the function of the button which deactivated the level page and activated the main menu page. The audio also pops out after the player clicks on the back button.

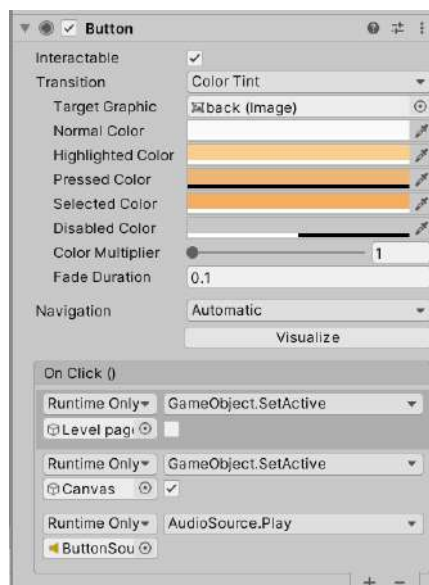


Figure 4.2.1.22: Inspector of the Back button of level page

Figure 4.2.1.23 shows the inspector of level management. It used to set the level selection in the level page. The level management included the script “LevelSelection”. There are six level select buttons that were inserted into Lvl Buttons, Level At is 1 and the Max_level is 6.

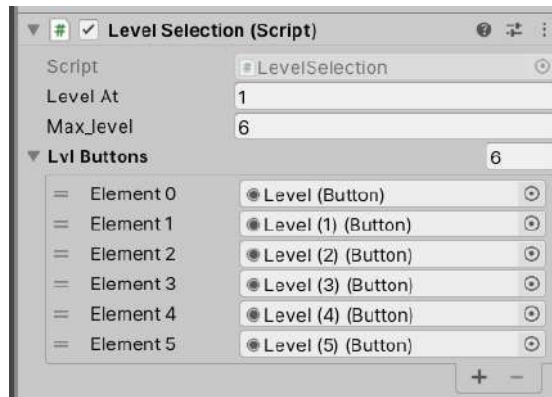


Figure 4.2.1.23: Inspector of the level management.

The figure 4.2.1.24 shows the LevelSelection Script that was inserted in the Level Management. In the Start(), LevelAt was set as 1. The Update() was used to set an interactable level button. The level button was interactable when the player passed the level. The reset() was used to reset the level button. The full script was shown in appendix 2.

```

12 // Start is called before the first frame update
13 void Start()
14 {
15     levelAt = 1;
16     levelAt = PlayerPrefs.GetInt("levelAt", 1);
17 }
18
19
20 private void Update()
21 {
22     for (int i = 0; i < lvlButtons.Length; i++)
23     {
24         if (i + 1 > levelAt)
25         {
26             lvlButtons[i].interactable = false;
27             Debug.Log(" " + levelAt);
28         }
29         else
30         {
31             lvlButtons[i].interactable = true;
32             Debug.Log(" " + levelAt);
33         }
34     }
35 }
36
37 public void Reset()
38 {
39     levelAt = 1;
40     PlayerPrefs.SetInt("levelAt", levelAt);
41 }
42
43 }
44

```

Figure 4.2.1.24: LevelSelection Script.

E. Pause Menu Page

The pause menu page pops out when the player presses on “ESC ” to pause the game application. In the figure 4.2.25, there are “RESUME” , “RESTART”, and “MAIN MENU” to let players choose. The function of “RESUME” is to play the game, “RESTART” is to restart the game level again and the “MAIN MENU” is to bring the player back to the main menu. After pressing the “ESC”, the game application also pauses.



Figure 4.2.25: Pause menu page

The figure 4.2.1.26 shows the script of Pause Menu. The pause menu script inserted into the pause menu in the game will pause when players press “ESC ” and this function was set by the “void Update()”. The Resume() was used to deactivate the pause menu and start the game by setting the GameIsPaused as false. Pause() was used to stop the time and all animation of the game. MainMenu() was used to bring the player to the main menu page after the player clicked on Main Menu. Quit() was used to exit the game. The full script was shown in appendix 3.

```

6 public class PauseMenu : MonoBehaviour
7 {
8     public static bool GameIsPaused = false;
9
10    public GameObject pauseMenuUI;
11
12    // Update is called once per frame
13    void Update()
14    {
15        if (Input.GetKeyDown(KeyCode.Escape))
16        {
17            if (GameIsPaused)
18            {
19                Resume();
20            }
21            else
22            {
23                Pause();
24            }
25        }
26    }
27    public void Resume()
28    {
29        pauseMenuUI.SetActive(false);
30        Time.timeScale = 1f;
31        GameIsPaused = false;
32    }
33
34    void Pause()
35    {
36        pauseMenuUI.SetActive(true);
37        Time.timeScale = 0f;
38        GameIsPaused = true;
39    }
40    public void MainMenu()
41    {
42        Time.timeScale = 1f;
43        SceneManager.LoadScene("Main Menu");
44    }
45    public void Quit()
46    {
47        Debug.Log("QUIT");
48        Application.Quit();
49    }

```

```

D:\Users\Acer\project_unity\PSM\Assets\Script\PauseMenu.cs 2
50 public void Restart()
51 {
52     SceneManager.LoadScene(SceneManager.GetActiveScene().name);
53     Time.timeScale = 1f;
54 }
55 }
56

```

Figure 4.2.1.26: Pause Menu Script.

F. Game Complete

Figure 4.2.1.27 shows the Game complete page and it will pop out when the player completes all of the levels of the game. There is an animation of the game complete and 2 buttons which are Main Menu and Quit button. The Main menu button brings the player to the main menu and the Quit button will exit the game. The script that use for the button are same as the Main Menu script.

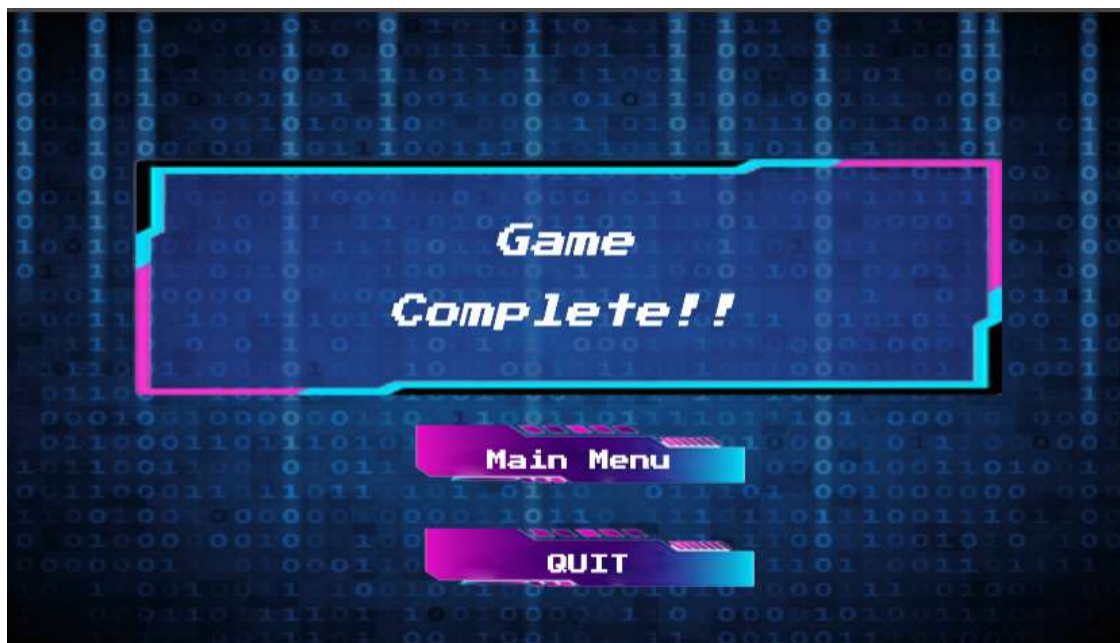


Figure 4.2.1.27: Game complete.

G. Interface of the game

Figure 4.2.1.28 shows the interface of the tutorial level, the top left corner shows the health bar of the player character, the medium part shows the requirement to pass the level and the top right corner shows the coin that the player has collected. The red health bar on the top left corner of the interface shows the health bar of the player character. If the player character collides with the enemy, the health bar will deduce. If the health bar becomes 0, players need to replay the level to pass the level. The requirement shown on the medium top of the game interface is the requirement of the coin collected to pass the level. The top right corner of the interface is the number of the coin that was collected by the player.



Figure 4.2.1.28: Interface of the tutorial

H. Dialog of the Npc in game



Figure 4.2.1.29: Dialog of the NPC

The NPC will follow the player to guide and chat with the player character. The dialog about the introduction of NPC will pop out as shown in the figure 4.2.1.29. The NPC

also will tell the player to avoid colliding with the enemy and so on. When the player character gets into the new environment, the NPC also will tell the player what the new enemies are and trap in the new environment.

Figure 4.2.1.30 shows the inspector of signbox. The dialog of the NPC was triggered when the player collided with the signbox. However, there also will trigger the dialog of the NPC in some place that is without anything. The box collider 2D was used as a box collider to trigger with the player.

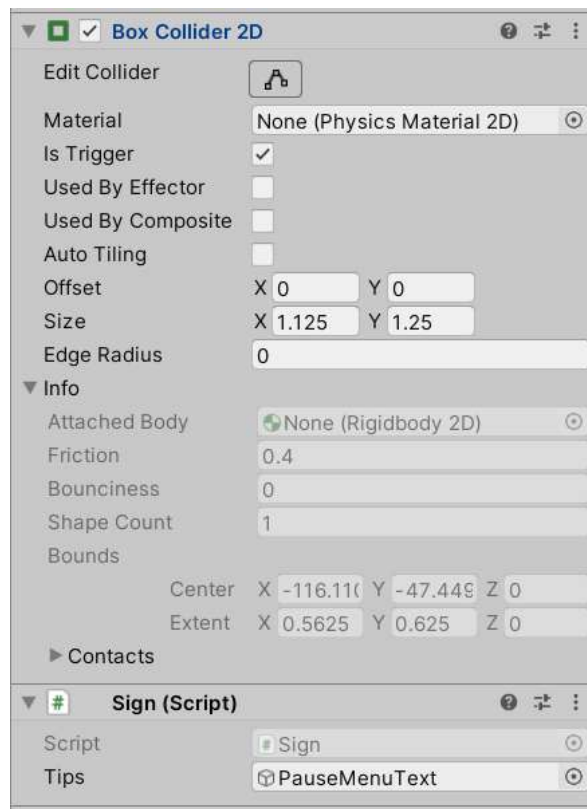


Figure 4.2.1.30: Inspector of Signbox.

Figure 4.2.1.31 shows the script of Signbox. The “OnTriggerEnter2D(Collider2D collision)” in the script was used to pop out the dialog when trigger the player character and the “OnTriggerExit2D(Collider collision)” was used to deactivated the dialog when the player was trigger exit the object or place. The dialog also destroys after the player exit collide with the box collider. Full script was shown in appendix 7.

```

10 private void OnTriggerEnter2D(Collider2D collision)
11 {
12     if (collision.tag == "Player")
13     {
14         Debug.Log("player in range");
15
16         Tips.SetActive(true);
17     }
18 }
19 private void OnTriggerExit2D(Collider2D collision)
20 {
21     if (collision.tag == "Player")
22     {
23         Debug.Log("player out of range");
24
25         Tips.SetActive(false);
26         Destroy(Tips);
27     }
28 }
29 }
30

```

Figure 4.2.1.31: Script of Sign.

The script in figure 4.2.1.32 was used to set the sign box without destroying the dialog and let the player review it. Full appendix was shown in appendix 29.

```

10 private void OnTriggerEnter2D(Collider2D collision)
11 {
12     if (collision.tag == "Player")
13     {
14         Debug.Log("player in range");
15
16         Tips.SetActive(true);
17     }
18 }
19 private void OnTriggerExit2D(Collider2D collision)
20 {
21     if (collision.tag == "Player")
22     {
23         Debug.Log("player out of range");
24
25         Tips.SetActive(false);
26     }
27 }
28 }
29

```

Figure 4.2.1.32: Script of SignNotDestroy

I. Loading Scene

When loading to the next scene, it took the same amount of time to render the asset of the next scene. So, the loading scene was used to load onto the next scene and let the player not feel so bored while waiting for the next scene. The figure 4.2.1.33 shows the loading scene of the game and the progress bar will process to let the player know the process of jumping to the next scene.

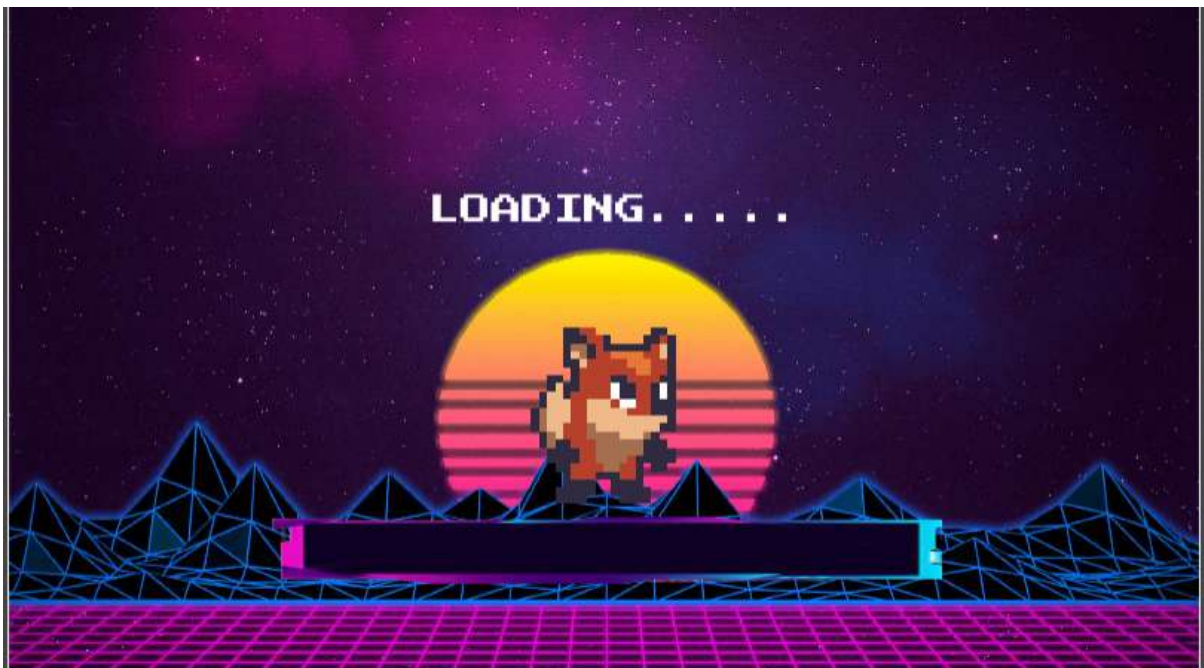


Figure 4.2.1.33: Loading Scene

The script in figure 4.2.1.34 was used to set the progress bar of the loading scene to show the progress of the jumping scene. The script will be duplicated to each level and changing class name and LoadSceneAsync() in Line 21. Full script was shown in appendix 30.

```

11 public void LoadScene()
12 {
13     StartCoroutine(LoadScene_Coroutine());
14 }
15
16 public IEnumerator LoadScene_Coroutine()
17 {
18     progressSlider.value = 0;
19     LoaderUI.SetActive(true);
20
21     AsyncOperation asyncOperation = SceneManager.LoadSceneAsync(1);
22     asyncOperation.allowSceneActivation = false;
23     float progress = 0;
24
25     while (!asyncOperation.isDone)
26     {
27         progress = Mathf.MoveTowards(progress, asyncOperation.progress,
28             Time.deltaTime);
29         progressSlider.value = progress;
30         if (progress >= 0.9f)
31         {
32             progressSlider.value = 1;
33             asyncOperation.allowSceneActivation = true;
34         }
35         yield return null;
36     }
37 }

```

Figure 4.2.1.34: Script of Loading Scene.

J. Game Over Panel

The panel in figure 4.2.1.35 will pop out when the life of the player character becomes 0. Players need to be careful to control the player character to avoid colliding with the enemy and trap.

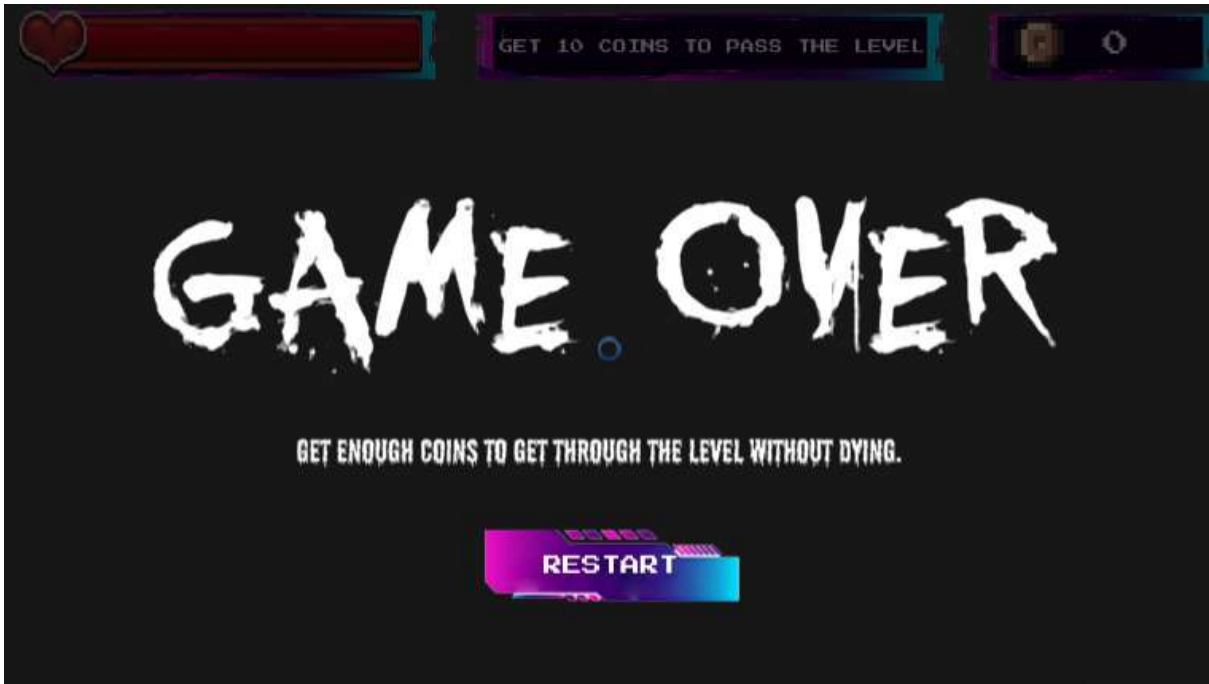


Figure 4.2.1.35: Game over panel

The figure 4.2.1.36 shows the button inspector of the game over panel. The button was used to restart the level when the player lost the game.

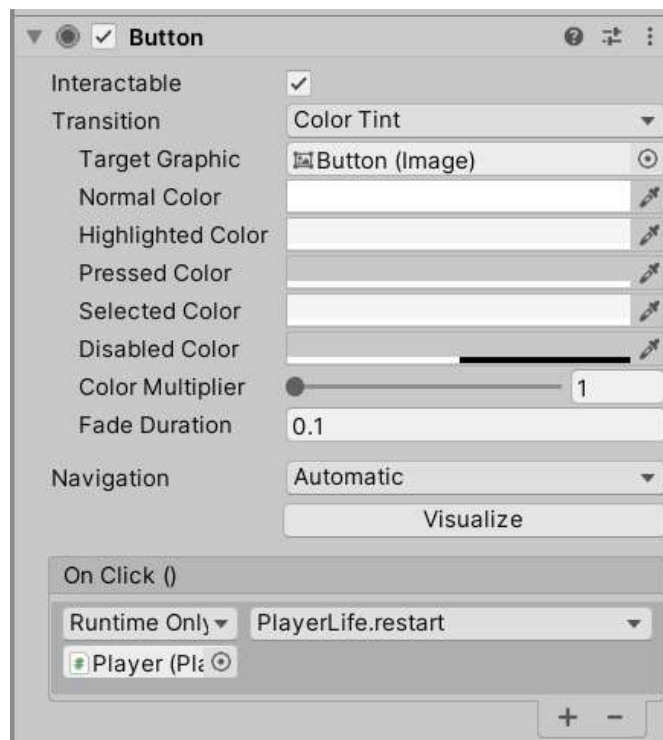


Figure 4.2.1.36: Inspector of the game over panel

K. Insufficient Coin Panel

The figure 4.2.1.37 shows the insufficient coins panel of the game. The panel will pop out when the player did not get enough coins and collide with the checkpoint. The restart button was used to restart the level.

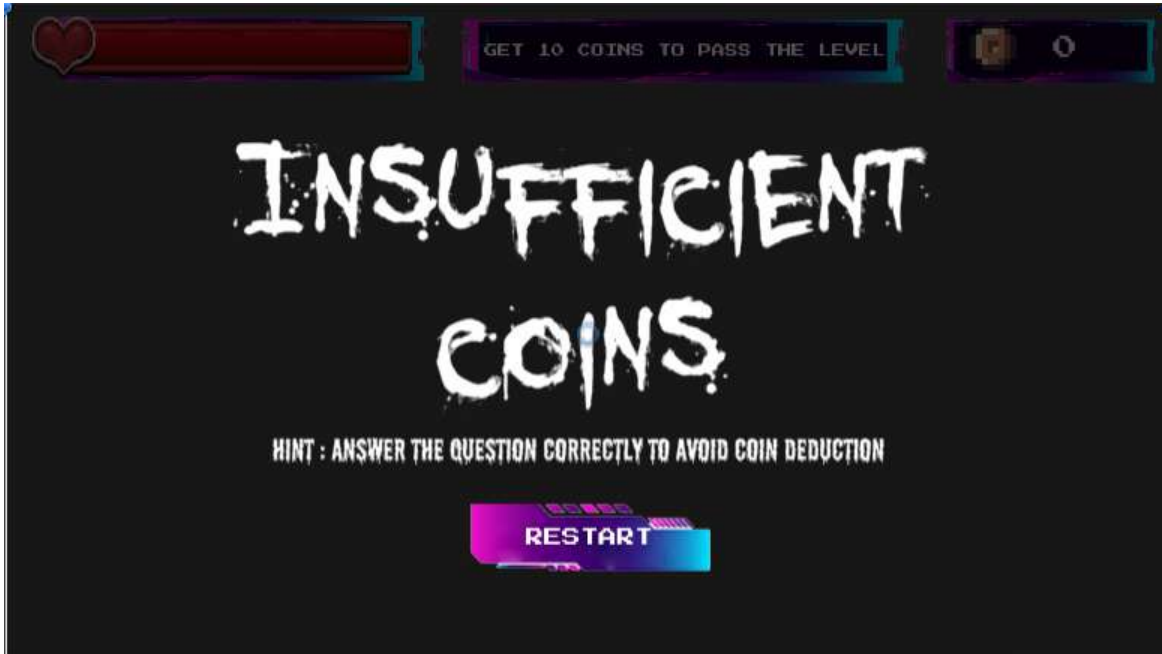


Figure 4.2.1.37: Insufficient coins panel

Inspector shown in figure 4.2.1.38 was used to set the button of the insufficient coins panel to restart the level.

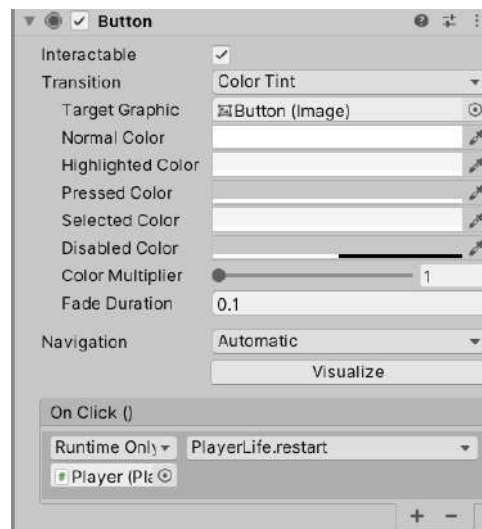


Figure 4.2.1.38: Inspector of the insufficient coins panel.

F. Game mechanic

A. Phishing Awareness Question

Figure 4.2.1.39 shows the question about phishing awareness after a player character collides with the computer in the game level. After colliding with the computer, the game also will pause to let the player have some time to answer the question. If a player answers wrongly, the button will become red, deduce 10 coins of the player character and show the number of the coin as “-10” as shown in the figure 4.2.1.40. If it is answered correctly, the button will become green as shown in figure 4.2.1.41. In the figure 4.2.1.42, the panel also disappears after waiting for a second. The computer also will disappear after the player did not trigger with the computer same as the figure 4.2.1.43.



Figure 4.2.1.39: Phishing Awareness Question Page



Figure 4.2.1.40: After answering the question wrongly.



Figure 4.2.1.41: After answering the question correctly.



Figure 4.2.1.42: The question panel disappears.



Figure 4.2.1.43: The computer disappears.

The script was inserted and the inspector will show as the figure 4.2.1.44. The inspector can set the number of the questions that want to be set in the game. After that, the question can be set with the choices and the answer of the question. The option part can be used to set the number of the option and the element of the option.

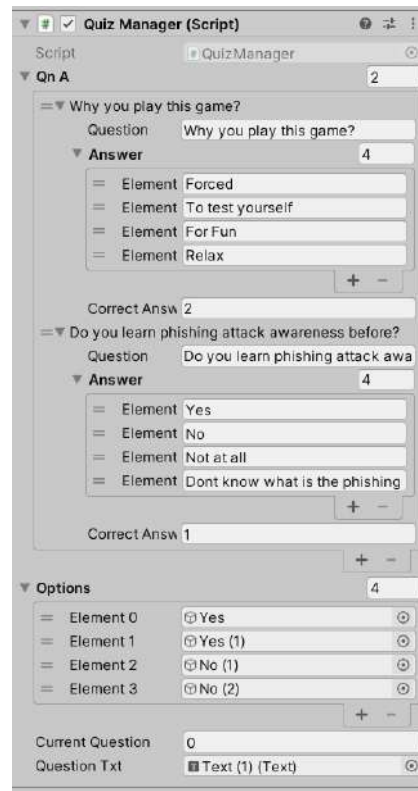


Figure 4.2.1.44: The inspector of Quiz Manager.

The script in figure 4.2.1.45 was used to set the question, answer, and correct answer of the game. The full script was shown in appendix 4.

```

...ect_unity\PSM\Assets\Script\Quiz\QuizAnswerAndQuestion.cs 1
1
2
3 [System.Serializable]
4
5 public class QuizAnswerAndQuestion
6 {
7     public string Question;
8     public string[] Answer;
9     public int CorrectAnswer;
10
11 }
12

```

Figure 4.2.1.45: Script of Quiz Answer and Question.

The script of answer in figure 4.2.1.46 was used to set the color of the correct answer and wrong answer. In “public void Answer()”, the color was set in green when the answer was correct and the color was red when the answer was wrong. The mark also was changed by using the code quizManager.correct and quizManager.wrong in QuizManager. The full script was shown in appendix 5.

```

6 public class AnswerScript : MonoBehaviour
7 {
8     public static bool Paused = false;
9     public bool isCorrect = false;
10    public QuizManager quizManager;
11
12    public Color startColor;
13
14    private void Start()
15    {
16        startColor = GetComponent<Image>().color;
17    }
18    public void Answer()
19    {
20
21        if(isCorrect)
22        {
23            GetComponent<Image>().color = Color.green;
24            Debug.Log("correct Answer");
25
26            // put add coin and correct panel
27            Time.timeScale = 1f;
28            Paused = false;
29
30            quizManager.correct();
31        }
32        else
33        {
34            GetComponent<Image>().color = Color.red;
35            Debug.Log("Wrong Answer");
36
37            // put deduce coin and wrong panel
38            Time.timeScale = 1f;
39            Paused = false;
40
41            quizManager.wrong();
42        }
43    }
44 }

```

Figure 4.2.1.46: Script of Answer

Figure 4.2.1.47 shows the Script of Quiz manager. The Start() was used to set the total number of questions and call the function of generateQuestion() to generate the question when start(). The correct() was used to add the number of coins, remove the current question, exit the question panel and jump to the next question after some seconds. The wrong() have the function of correct() but the coin will be deducted. The Wait SetAnswer() was used to set the time to jump to the next question and the SetAnswer() was used to set the answer and question. The generateQuestion() was used to generate the next question randomly. The full script was shown in appendix 6.

```

17 private void Start()
18 {
19     totalQuestion = QnA.Count;
20     generateQuestion();
21 }
22
23 public void correct()
24 {
25     FindObjectOfType<itemCollect>().Increasecoin();
26     QnA.RemoveAt(currentQuestion);
27     FindObjectOfType<popOut>().PopExit();
28     StartCoroutine(WaitForNext());
29 }
30
31 public void wrong()
32 {
33     FindObjectOfType<itemCollect>().Deducecoin();
34     QnA.RemoveAt(currentQuestion);
35     FindObjectOfType<popOut>().PopExit();
36     StartCoroutine(WaitForNext());
37 }
38
39 IEnumerator WaitForNext()
40 {
41     yield return new WaitForSeconds(2);
42     generateQuestion();
43 }
44
45 void SetAnswers()
46 {
47     for(int i = 0; i < options.Length; i++)
48     {
49         options[i].GetComponent<Image>().color = options
50
51         [i].GetComponent<AnswerScript>().startColor;
52         options[i].GetComponent<AnswerScript>().isCorrect = false;
53         //WrongPanel();
54         options[i].transform.GetChild(0).GetComponent<Text>().text = QnA
55         [currentQuestion].Answer[i];
56
57         if(QnA[currentQuestion].CorrectAnswer == i+1)
58         {
59             options[i].GetComponent<AnswerScript>().isCorrect = true;
60             options[i].GetComponent<Image>().color = options
61             [i].GetComponent<AnswerScript>().startColor;
62             //CorrectPanel();
63         }
64     }
65 }
66
67 void generateQuestion()
68 {
69     if (QnA.Count > 0)
70     {
71         currentQuestion = Random.Range(0, QnA.Count);
72
73         QuestionTxt.text = QnA[currentQuestion].Question;
74         FindObjectOfType<popOut>().PopExit();
75         SetAnswers();
76     }
77     else
78     {
79         Debug.Log("Out of Question"); //
80         FindObjectOfType<popOut>().PopExit();
81     }
82 }

```

Figure 4.2.1.47: Script of Quiz Manager.

The computer was used to pop out the question when the player triggered it. The inspector in figure 4.2.1.48 was used to set the box collider and a pop out script was used to pop out the question and disappear the computer.



Figure 4.2.1.49: Inspector of computer

The “OnTriggerEnter2D(Collider2D other)” in figure 4.2.1.50 was used to pop out the question, pause the screen when the player character triggers with the computer. The “OnTriggerExit2D(Collider2D other)” was used to disappear the computer when the player character triggered out of the range of the computer's box collider. Full script was shown in appendix 8.

```

21 public void OnTriggerEnter2D(Collider2D other)
22 {
23
24     if (other.tag == "Player")
25     {
26
27         Time.timeScale = 0f;
28         Paused = true;
29         popUpBox.SetActive(true);
30         Debug.Log("computer is triggered");
31
32     }
33     else
34     {
35         popUpBox.SetActive(false);
36
37     }
38
39 }
40
41 public void OnTriggerExit2D(Collider2D other)
42 {
43     Destroy(computer);
44 }
45
46 public void PopExit()
47 {
48     StartCoroutine(WaitForWhile());
49 }

```

```

50
51 IEnumerator WaitForWhile()
52 {
53     yield return new WaitForSeconds(1);
54     popUpBox.SetActive(false);
55 }
56 }
57

```

Figure 4.2.1.50: Script of pop out.

B. Enemies and Traps in different levels

There are different types of enemies and traps for each 2 levels. Figure 4.2.1.51 shows the enemies and traps in tutorial and Level 2. If a player collides with the enemies and traps too many times, the level will restart when the health bar of the player character becomes 0. For the enemies and traps in figure 4.2.1.52, there are different enemies and traps in the new environment such as the blue virus, saw, green virus, arrow and blue fire. The green virus of the figure 4.2.1.52 will chase the player character if the player character passes through it after inserting the script “chasePlayer”. For the figure 4.2.1.53, there are also different types of enemies and traps for the new environment “Neon”. All of the moving enemies or traps were using the script name “EnemyAI” to make the enemies move according to the waypoints that were set in the game application. All of the enemies and traps have inserted the script name “Enemy” to deduce the player character health bar when the player character collides on it. For the square in the figure 4.2.1.53, some of the squares will rotate in clockwise and anti-clockwise. The arrow and the laser that show in figure 4.2.1.52 and figure 4.2.1.53 were using the scripts name “EnemyProjectile” and “ArrowTrap”. The arrow will automatically shoot out and the speed and the time to reset the arrow can be set in the inspector. For the figure 4.2.1.53, there are some of the purple neon platforms which are falling platforms. The player character needs to jump faster to prevent falling into the traps. Some of the enemies and traps also have set the animation to let them be animated. All of the traps and enemies have put the box collide to collide with the player character.



Figure 4.2.1.51: Enemies and traps in tutorial and Level 2.

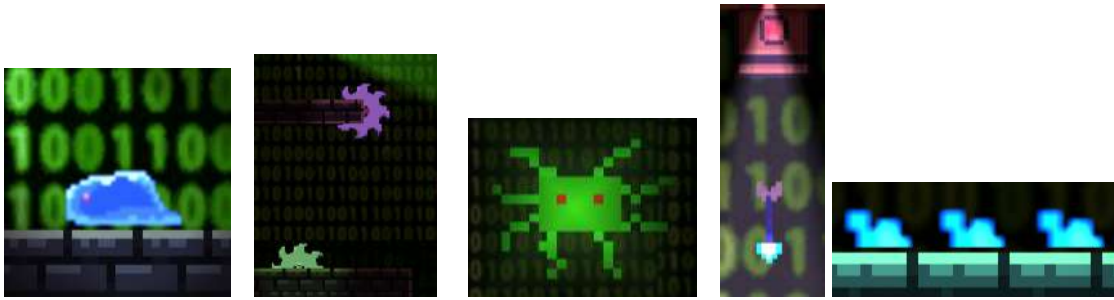


Figure 4.2.1.52: Enemies and traps in Level 3 and Level 4.



Figure 4.2.1.53: Enemies and traps in Level 5 and 6.

Figure 4.2.1.54 shows the enemies move using the following waypoints. The enemies and traps move by inserting the script Waypoint.



Figure 4.2.1.54: Enemies and traps move following the waypoints.

The inspector in figure 4.2.1.55 shows an inspector of enemies and traps that can move. The box collider 2D was used to trigger with the player character and deduce the health bar of the player character by using enemy script. The enemies and the traps can follow the waypoint by inserting the script Enemy AI.

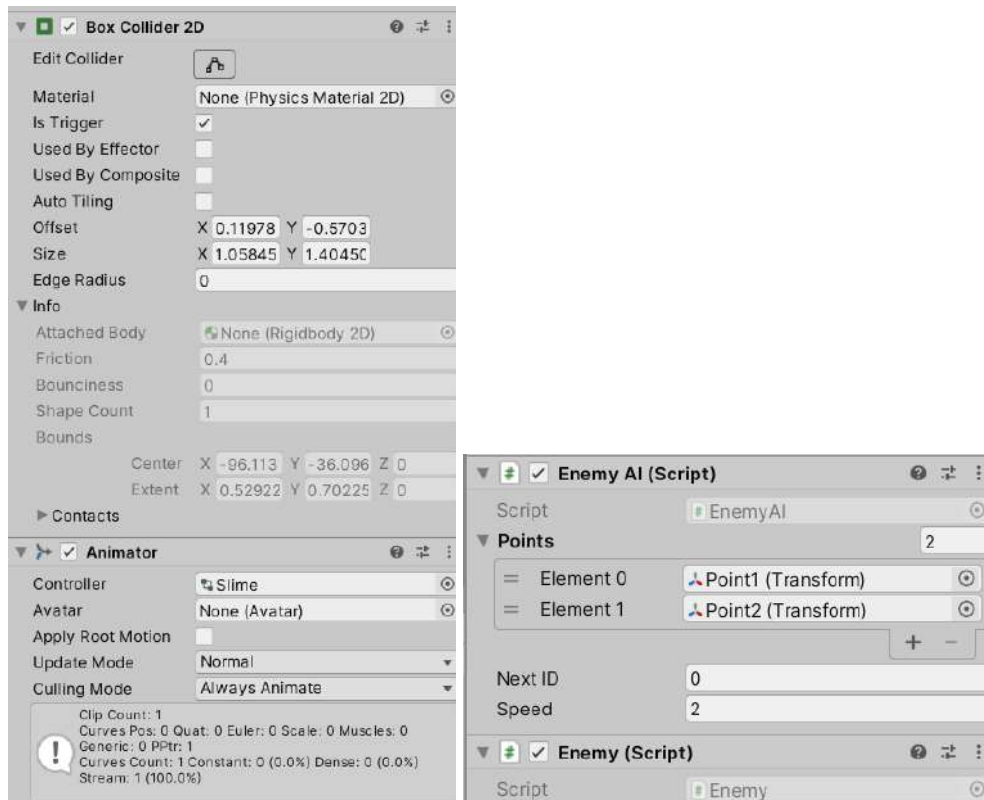


Figure 4.2.1.55: Inspector of enemies and traps that can move.

Figure 4.2.1.56 shows the script of the enemy and was used to deduce 20 percent life of the player character when triggered with the enemies and traps. Full script was shown in appendix 10.

```

7     protected int decayAmount = 20;
8
9
10
11    private void Start()
12    {
13    }
14    private void Reset()
15    {
16        GetComponent<BoxCollider2D>().isTrigger = true;
17    }
18
19    protected void OnTriggerEnter2D(Collider2D collision)
20    {
21        if (collision.tag == "Player")
22        {
23            Debug.Log($"{name} Triggered");
24            FindObjectOfType<HealthBar>().LoseHealth(decayAmount);
25        }
26    }
27    }
28 }
29
30 }
31

```

Figure 4.2.1.56: Script of Enemy

The script of enemy AI that is shown in figure 4.2.1.57 was used to let the enemies and traps move following the waypoints one by one and loop it. Full script was shown in appendix 11.

```
17 void Init()
18 {
19     //Make box collider trigger
20     GetComponent<BoxCollider2D>().isTrigger = true;
21
22     //Create Root object
23     GameObject root = new GameObject(name + "_Root");
24     //Reset Position of Root to enemy object
25     root.transform.position = transform.position;
26     //Set enemy object as child of root
27     transform.SetParent(root.transform);
28     //Create Waypoint onject
29     GameObject waypoints = new GameObject("Waypoints");
30     //Reset Waypoint position to root
31     //Make waypoint object child of root
32     waypoints.transform.SetParent(root.transform);
33     waypoints.transform.position = root.transform.position;
34     //Create two points (game object) and reset their position to waypoints
35     //objects
36     //Make the points childern of waypoint object
37     GameObject p1 = new GameObject("Point1"); p1.transform.SetParent
38         (waypoints.transform); p1.transform.position = Vector3.zero;
39     GameObject p2 = new GameObject("Point2"); p2.transform.SetParent
40         (waypoints.transform); p2.transform.position = Vector3.zero;
41
42     //Init points list then add the points to it
43     points = new List<Transform>();
44     points.Add(p1.transform);
45     points.Add(p2.transform);
46 }
47
48 private void Update()
49 {
```

```

47     MoveToNextPoint();
48 }
49 void MoveToNextPoint()
50 {
51     //Get the next point transform
52     Transform goalPoint = points[nextID];
53     //Flip the enemy towards to look into the point's direction
54     if (goalPoint.transform.position.x > transform.position.x)
55         transform.localScale = new Vector3(-1, 1, 1);
56     else
57         transform.localScale = new Vector3(1, 1, 1);
58     //Move the enemy towards to goal point
59     transform.position = Vector2.MoveTowards(transform.position,
60         goalPoint.position, speed * Time.deltaTime);
61     //Check the distance between enemy and goal point to trigger next point
62     if (Vector2.Distance(transform.position, goalPoint.position) < 0.5f)
63     {
64         //check if we are at the end of the line (make the change -1)
65         if (nextID == points.Count - 1)
66             idChangeValue = -1;
67         //check if we are at the start of the line (make the change +1)
68         if (nextID == 0)
69             idChangeValue = 1;
70         //Apply the change on the nextID
71         nextID += idChangeValue;
72         //nextID = nextID + idChangeValue
73     }
74 }
75 }
76

```

Figure 4.2.1.57: Script of Enemy AI

Figure 4.2.1.58 shows the traps which are not moving. The traps were deduced from the life of the player character when the player character collides with the traps.

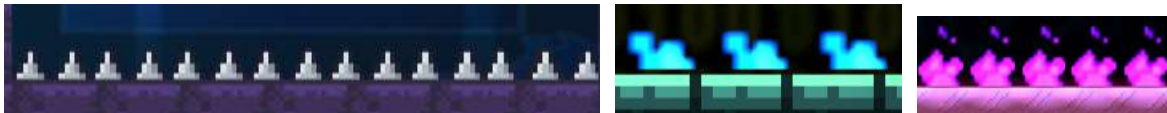


Figure 4.2.1.58: Traps without moving.

The inspector in figure 4.2.1.59 was used to help to set the function of the traps. The box collider 2D was used to trigger with the player character. Some traps did not tick the “Is Trigger” box in the box collider 2D and the player character will die and restart the level. When the “Is Trigger” was ticked, the life of the player character was deducted and restarted the level when the life of the player character became 0.

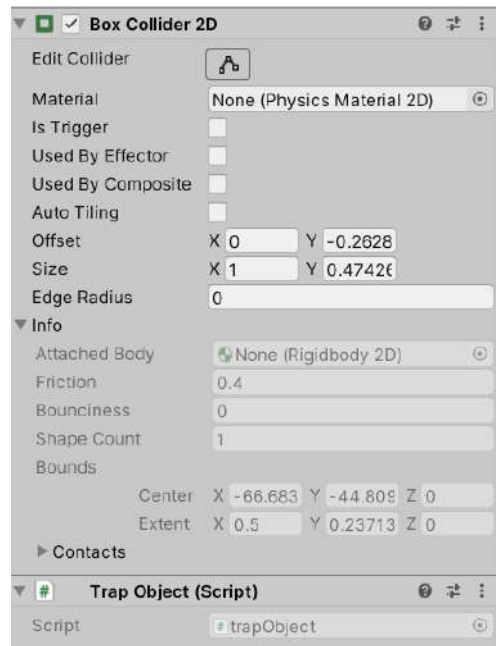


Figure 4.2.1.59: Inspector of the traps

The script in figure 4.2.1.60 was used to deduce the life of the player character when the player character collided with the trap and this function was set in OnTriggerEnter2D(). Full script was shown in appendix 12.

```

5 public class trapObject : MonoBehaviour
6 {
7     int decayAmount = 20;
8     private void Reset()
9     {
10        GetComponent<BoxCollider2D>().isTrigger = true;
11    }
12
13    private void OnTriggerEnter2D(Collider2D collision)
14    {
15        if (collision.tag == "Player")
16        {
17            Debug.Log($"{name} Triggered");
18            FindObjectOfType<HealthBar>().LoseHealth(decayAmount);
19        }
20    }
21 }
22
  
```

Figure 4.2.1.60: Script of traps.

The figure 4.2.1.61 was a virus in level 3 and 4. The virus acted as an enemy to chase the player character when the player character was in the range of chase of the enemy.



Figure 4.2.1.61: Virus enemy

The inspector of Virus enemy that shows in figure 4.2.1.62 was used to set the function of virus enemy. The box collider 2D was used to trigger with the player and Chase Player was used to chase the player character. Rigidbody 2D also was set for the virus enemy.

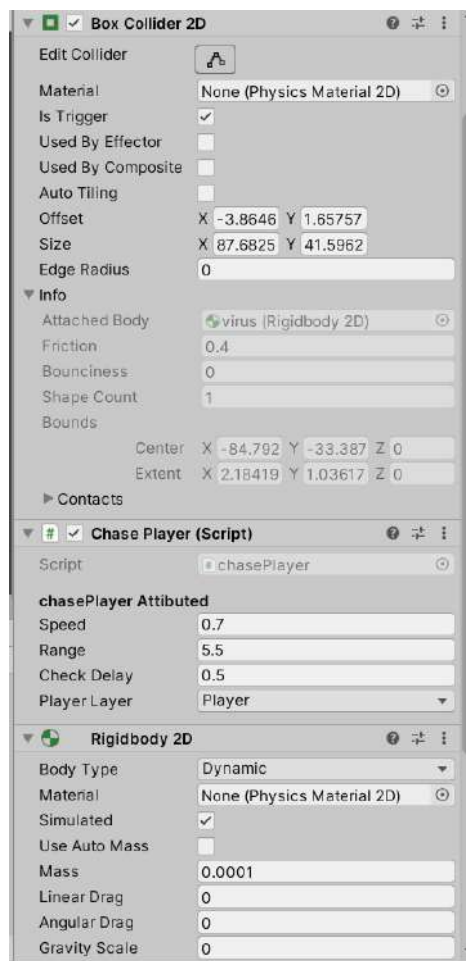


Figure 4.2.1.62: Inspector of virus enemy

The script in figure 4.2.1.63 was inserted into the virus enemy. The virus enemy will chase the player when the player is in the range and the virus enemy will stop chasing the player character when the player character is out of the range. Full script was shown in appendix 13.

```

25 void Update()
26 {
27     //Move computer to destination only if attacking
28     if (attacking)
29         transform.Translate(destination * Time.deltaTime * speed);
30     else
31     {
32         checkTimer += Time.deltaTime;
33         if (checkTimer > checkDelay)
34             CheckForPlayer();
35     }
36 }
37
38 private void CheckForPlayer()
39 {
40     CalculaterDirections();
41
42     //Check if computer sees player
43     for (int i = 0; i < directions.Length; i++)
44     {
45         Debug.DrawRay(transform.position, directions[i], Color.red);
46         RaycastHit2D hit = Physics2D.Raycast(transform.position, directions
47             [i], range, playerLayer);
48         if(hit.collider != null && ! attacking)

```

```

...Acer\project_unity\PSM\Assets\Script\Enemy\chasePlayer.cs
49     {
50         attacking = true;
51         destination = directions[i];
52         checkTimer = 0;
53     }
54 }
55 }
56
57 private void CalculaterDirections()
58 {
59     directions[0] = transform.right * range; //right direction
60     directions[1] = -transform.right * range; //left direction
61     directions[2] = transform.up * range; //Up direction
62     directions[3] = -transform.up * range; //down direction
63 }
64
65 private void stop()
66 {
67
68     destination = transform.position; //set destination as current position
69     attacking = false;
70 }
71
72
73 private void OnTriggerEnter2D(Collider2D collision)
74 {
75     if(collision.tag == "Player")
76     {
77         FindObjectOfType<HealthBar>().LoseHealth(decayAmount);
78     }
79     // stop once when hits something
80     stop();
81 }
82 }
83 }

```

Figure 4.2.1.63: Script of Chase Player

The script in figure 4.2.1.64 was used to set the function when the player character triggers with the box collider, the virus enemy will reset the position back to the original position. Full script was shown in appendix 14.

```

10 private void Awake()
11 {
12     //save the initial position of the enemies
13     initialPosition = new Vector3[enemies.Length];
14     for(int i = 0; i < enemies.Length; i++)
15     {
16         if(enemies[i] != null)
17             initialPosition[i] = enemies[i].transform.position;
18     }
19 }
20 public void ActivateRoom(bool _status)
21 {
22     //Activate/deactivate enemies
23     for(int i = 0; i < enemies.Length; i++)
24     {
25         if(enemies[i] != null)
26         {
27             enemies[i].SetActive(_status);
28             enemies[i].transform.position = initialPosition[i];
29         }
30     }
31 }
32 private void OnTriggerEnter2D(Collider2D collision)
33 {
34     if (collision.tag == "Player")
35     {
36         ActivateRoom(true);
37     }
38     else
39     {
40         ActivateRoom(false);
41     }
42 }
43 }
44

```

Figure 4.2.1.64: Script of reset position.

The arrow traps in figure 4.2.1.65 are the traps of the game and players need to avoid colliding with it to prevent deduced life.

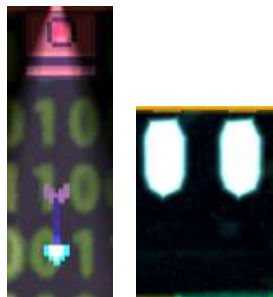


Figure 4.2.1.65: Arrow trap.

The figure 4.2.1.66 shows the inspector of arrow traps to set the function of the arrow traps. The arrows in the inspector were used to set the number of arrows and the fire point was set to let the arrow fire from it. It functioned as an arrow trap management to control the arrows.

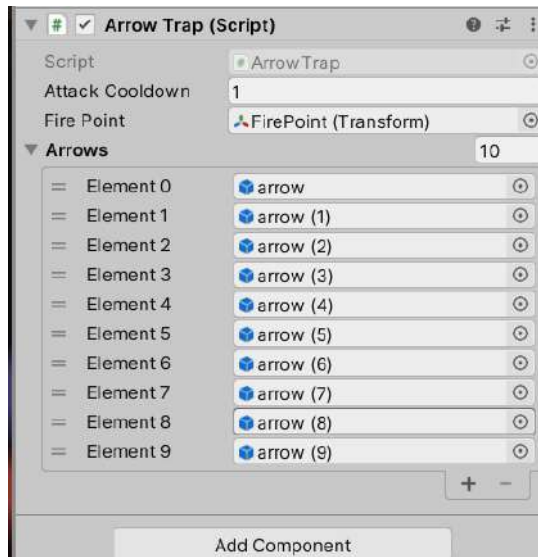


Figure 4.2.1.66: Inspector of arrow traps.

The script of arrow trap shown in figure 4.2.1.67 was used to set the firepoint, number of arrows and attack cooldown. Full script was shown in appendix 15.

```

D:\Users\Acer\project_unity\PSM\Assets\Script\Arrow\ArrowTrap.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ArrowTrap : MonoBehaviour
6 {
7     [SerializeField] private float attackCooldown;
8     [SerializeField] private Transform firePoint;
9     [SerializeField] private GameObject[] arrows;
10    private float cooldownTimer;
11
12    private void Attack()
13    {
14        cooldownTimer = 0;
15
16        arrows[FindArrow()].transform.position = firePoint.position;
17        arrows[FindArrow()].GetComponent<EnemyProjectile>().ActiveProjectile();
18    }
19    private int FindArrow()
20    {
21        for (int i = 0; i < arrows.Length; i++)
22        {
23            if (!arrows[i].activeInHierarchy)
24                return i;
25        }
26        return 0;
27    }
28    private void Update()
29    {
30        cooldownTimer += Time.deltaTime;
31
32        if (cooldownTimer >= attackCooldown)
33            Attack();
34    }
35 }
36 }
37
38

```

Figure 4.2.1.67: Script of Arrow Trap.

Figure 4.2.1.68 shows the inspector of arrows and it is used to set the function of the arrows. The arrows insert box collider 2D, rigidbody 2D and script enemy projectile. It used to set the speed and reset time of arrows. The box collider was used to trigger with the player character and deduced the life of the player character.

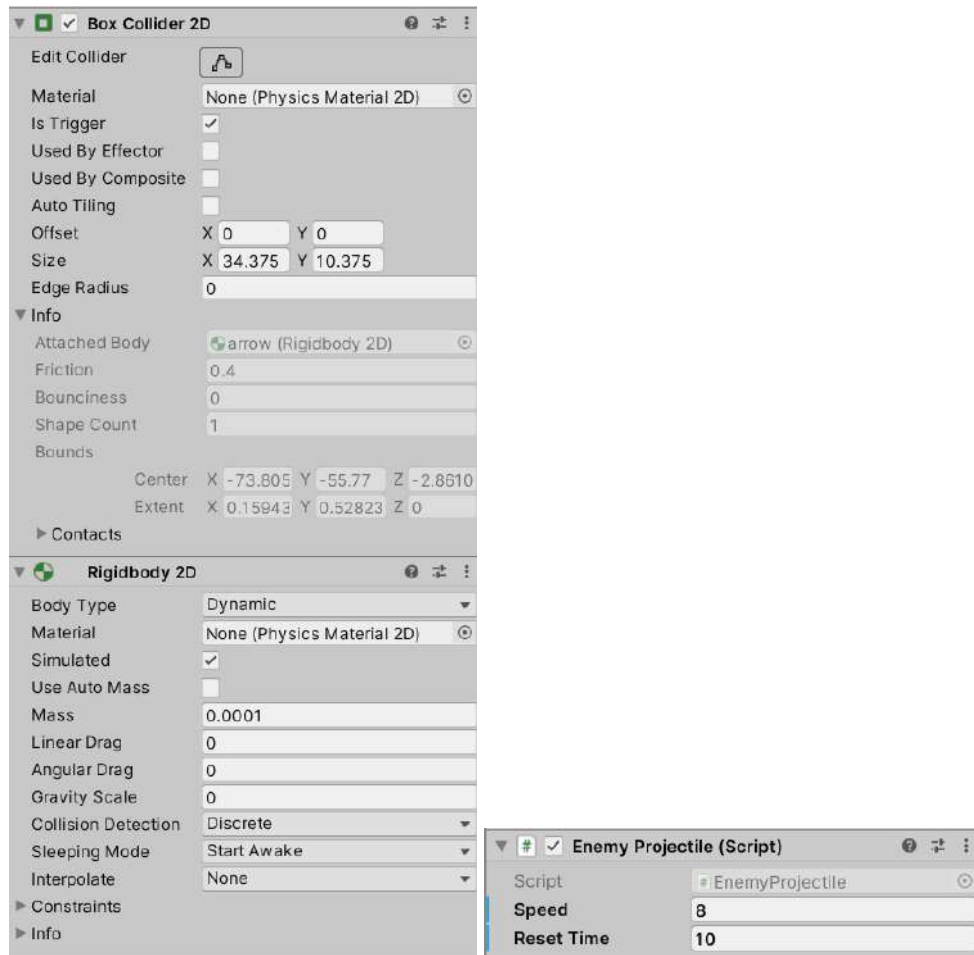


Figure 4.2.1.68: Inspector of arrow.

The figure 4.2.1.69 shows the script of the enemy projectile. It was used to set the speed and the reset time of the arrows. Full script was shown in appendix 16.

```

13 public void ActiveProjectile()
14 {
15     lifeTime = 0;
16     gameObject.SetActive(true);
17 }
18
19 private void Update()
20 {
21     float movementSpeed = speed * Time.deltaTime;
22     transform.Translate(movementSpeed, 0, 0);
23
24     lifeTime += Time.deltaTime;
25     if (lifeTime > resetTime)
26         gameObject.SetActive(false);
27 }
28
29 private void OnTriggerEnter2D(Collider2D collision)
30 {
31     base.OnTriggerEnter2D(collision); //execute logic from parent script
32     first
33     gameObject.SetActive(false); // when this hits any objects deactivate
34     arrow
35 }
36

```

Figure 4.2.1.69: Script of Enemy projectile.

The figure 4.2.1.70 was a falling platform and it was one of the traps. The platform will fall after a few seconds and players need to jump faster to prevent falling into some traps.



Figure 4.2.1.70: Falling platform.

The figure 4.2.1.71 is a script that is used to set the platform fall when the player character jumps on it after a few seconds. The platform also will move back to its original location after a few seconds.

```

19 void Update()
20 {
21     if (platformMovingBack)
22         transform.position = Vector2.MoveTowards(transform.position,
23             initialPosition, 20f * Time.deltaTime);
24
25     if (transform.position.y == initialPosition.y)
26         platformMovingBack = false;
27 }
28 void OnCollisionEnter2D(Collision2D col)
29 {
30     if (col.gameObject.name.Equals("Player") && !platformMovingBack)
31     {
32         Invoke("DropPlatform", 0.2f);
33     }
34 }
35
36 void DropPlatform()
37 {
38     rb.isKinematic = false;
39     Invoke("GetPlatformBack", 1.5f);
40 }
41
42 void GetPlatformBack()
43 {
44     rb.velocity = Vector2.zero;
45     rb.isKinematic = true;
46     platformMovingBack = true;
47 }
48

```

Figure 4.2.1.71: Script of falling platform

The cubes in the figure 4.2.1.71 were rotated and the player needed to jump on it to pass the level. The cubes were rotated by inserting the animation to the cubes that show in the inspector of figure 4.2.1.72.



Figure 4.2.1.71: Rotating cube.

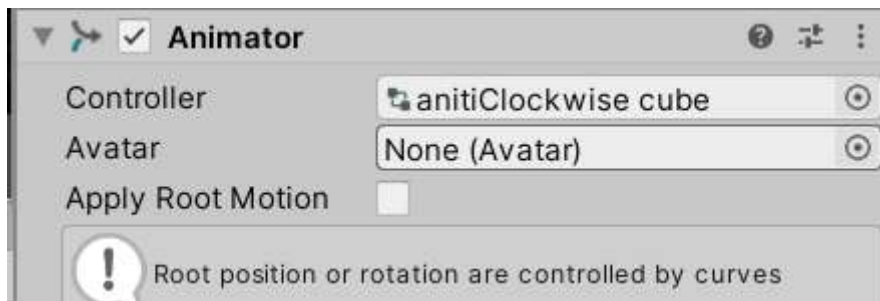


Figure 4.2.1.72. Inspector of cubes.

C. Moving Platform and ladder

The function of the moving platform and ladder were used to move from one place to prevent colliding with traps or enemies or climb up to the higher place as shown in the figure 4.2.1.73. The “climb” script also inserts on the ladder to let the player character climb on it. The “StickyPlatform” script and “wayPoint” script insert to the moving platform to let the player character stick on it and let the moving platform move.



Figure 4.2.1.73: The moving platform and ladder

Figure 4.2.1.74 showed the three box collider 2D which are at the top, bottom of the ladder and whole ladder. The inspector was used to set the function of the box collider 2D.

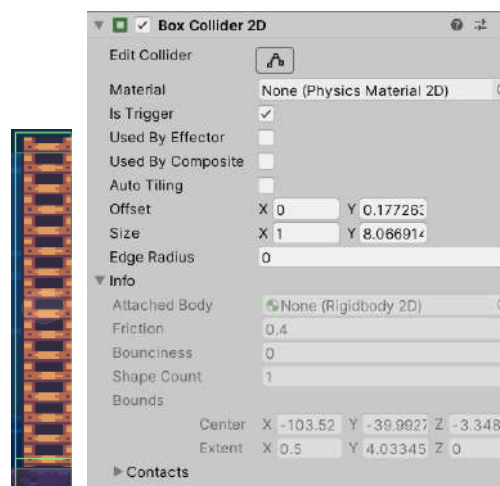


Figure 4.2.1.74: Three box collider 2D and inspector.

The figure 4.2.1.75 shows the inspector of the moving platform and it used to set the function of the moving platform. The box collider 2D was used to let the player character stand on it and the waypoint was to let the moving platform move following the waypoint.

The sticky platform was used to let the player character stick on the platform and the animator was used to set the animation of the moving platform.

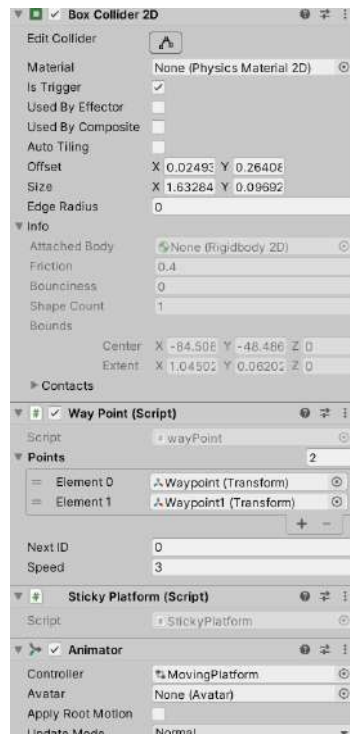


Figure 4.2.1.75: Inspector of moving platform

The figure 4.2.1.76 shows the script of the waypoint and it is used to set the waypoint of the moving platform. It is also used to set the speed of the moving platform. Full script was shown in appendix 9.

```

5 public class WaypointNotFlip : MonoBehaviour
6 {
7     [SerializeField] private GameObject[] waypoints;
8     private int currentWaypointIndex = 0;
9
10    [SerializeField] private float speed = 2f;
11
12    private void Update()
13    {
14        if (Vector2.Distance(waypoints[currentWaypointIndex].transform.position,
15            transform.position) < .1f)
16        {
17            currentWaypointIndex++;
18            if (currentWaypointIndex >= waypoints.Length)
19            {
20                currentWaypointIndex = 0;
21            }
22        }
23        transform.position = Vector2.MoveTowards(transform.position, waypoints
24            [currentWaypointIndex].transform.position, Time.deltaTime * speed);
25    }
26 }

```

Figure 4.2.1.76: Script of waypoint.

D. Player movement

The figure 4.2.1.77 shows the inspector of the player character. The inspector of player characters was inserted with a box collider 2D to trigger with other assets, animator to set the animation of the player character when in different movements, player movement, player life, health bar, item collect, climb and different audio source for different situations.

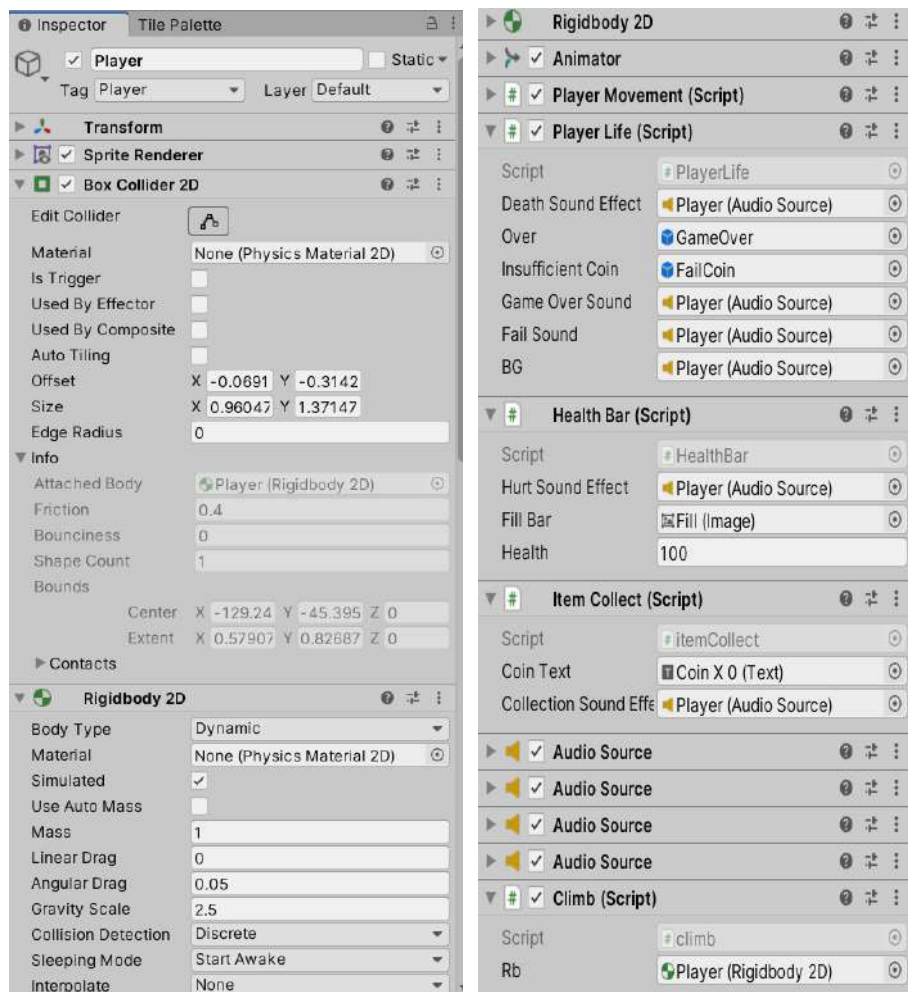


Figure 4.2.1.77: Inspector of player character.

The figure 4.2.1.78 shows the script of the climb. The script was used to let the player character climb the ladder after pressing “W” and set the animation of the player character when climbing. Full script was in the appendix 18.

```

14 void Update()
15 {
16     anim = GetComponent<Animator>();
17     vertical = Input.GetAxis("Vertical");
18
19     if (isLadder && Mathf.Abs(vertical) > 0f)
20     {
21         isClimbing = true;
22     }
23 }
24
25
26 private void FixedUpdate()
27 {
28     if (isClimbing)
29     {
30         rb.gravityScale = 2f;
31         rb.velocity = new Vector2(rb.velocity.x, vertical * speed);
32     }
33     else
34     {
35         rb.gravityScale = 4f;
36     }
37 }
38
39 private void OnTriggerEnter2D(Collider2D collision)
40 {
41     if (collision.CompareTag("Ladder"))
42     {
43         isLadder = true;
44         isClimbing = true;
45         //anim.SetBool("climb", true);
46     }
47 }
48
49 private void OnTriggerExit2D(Collider2D collision)
50 {
51     if (collision.CompareTag("Ladder"))
52     {
53         isLadder = false;
54         isClimbing = false;
55         //anim.SetBool("climb", false);
56     }
57 }
58 }

```

Figure 4.2.1.78: Script of Climb.

The figure 4.2.1.79 showed the script of player movement and it used to set the animation of the player character and movement. Full script was shown in appendix 20.


```

38 public void Update()
39 {
40     dirX = Input.GetAxis("Horizontal");
41
42     rb.velocity = new Vector2(dirX * moveSpeed, rb.velocity.y);
43     if (Input.GetButtonUp("Jump") && IsGrounded())
44     {
45         jumpSoundEffect.Play();
46         rb.velocity = new Vector2(rb.velocity.x, jumpForce);
47     }
48     UpdateAnimationState();

53 private void UpdateAnimationState()
54 {
55
56     if (dirX > 0f)
57     {
58         state = MovementState.running;
59         sprite.flipX = false;
60     }
61     else if (dirX < 0f)
62     {
63         state = MovementState.running;
64         sprite.flipX = true;
65     }
66     else
67     {
68         state = MovementState.idle;
69     }
70
71     if (rb.velocity.y > .1f)
72     {
73         state = MovementState.jumping;
74     }
75     else if (rb.velocity.y < -.1f)
76     {
77         state = MovementState.falling;
78     }
79
80     anim.SetInteger("state", (int)state);
81 }
82
83 private bool IsGrounded()
84 {
85     return Physics2D.BoxCast(coll.bounds.center, coll.bounds.size, 0f,
86         Vector2.down, .1f, jumpableGround);
87 }
88 }
89

```

Figure 4.2.1.80: Script of Player movement.

The figure 4.2.1.81 was shown the script of player life and it used to let the player character play the die animation when the health bar became 0. The game also will restart when the health bar becomes 0 and trigger an audio when die. The score of coins collected also will reset. The insufficient panel will pop out when the player collected not enough coins and the game over panel will pop out when the life of the player character becomes 0. Full script was shown in appendix 21.

```

6 public class PlayerLife : MonoBehaviour
7 {
8     private Rigidbody2D rb;
9     private Animator anim;
10
11     [SerializeField] private AudioSource deathSoundEffect;
12     public GameObject Over;
13     public GameObject InsufficientCoin;
14     [SerializeField] private AudioSource GameOverSound;
15     [SerializeField] private AudioSource FallSound;
16     [SerializeField] private AudioSource BG;
17
18     // Start is called before the first frame update
19     private void Start()
20     {
21         rb = GetComponent<Rigidbody2D>();
22         anim = GetComponent<Animator>();
23     }
24
25     private void OnCollisionEnter2D(Collision2D collision)
26     {
27         if (collision.gameObject.CompareTag("Trap"))
28         {
29             Die();
30         }
31     }
32
33     private void OnTriggerEnter2D(Collider2D collision)
34     {
35         if (collision.gameObject.CompareTag("Trap"))
36         {
37             Die();
38         }
39     }
40
41     public void Die()
42     {
43     }
44
45     deathSoundEffect.Play();
46     rb.bodyType = RigidbodyType2D.Static;
47     anim.SetTrigger("death");
48     resetScore();
49 }

```

```

...r\project_unity\PSM\Assets\Script\(\D)Player\PlayerLife.cs
50 public void GameOver()
51 {
52     //1- Restart the scene
53     Over.SetActive(true);
54     Time.timeScale = 0f;
55     GameOverSound.Play();
56     BG.Pause();
57     //2- Reset the player's position
58     //Save the player's initial position when game starts
59     //When respawning simply reposition the player to that init position
60 }
61
62 public void insufficientCoin()
63 {
64     InsufficientCoin.SetActive(true);
65     Time.timeScale = 0f;
66     FallSound.Play();
67     BG.Pause();
68 }
69
70 public void restart()
71 {
72     SceneManager.LoadScene(SceneManager.GetActiveScene().name);
73     resetScore();
74     Time.timeScale = 1f;
75 }
76
77 public void resetScore()
78 {
79     itemCollect.COIN = 0;
80     PlayerPrefs.SetInt("score", itemCollect.COIN);
81 }
82
83 }
84

```

Figure 4.2.1.81: Script of Player Life

The figure 4.2.1.82 was shown a script of a health bar and it was used to set the life of the player character and trigger an audio when the player character collides with obstacles. Full script shown in appendix 22.

```

15     public void LoseHealth(int value)
16     {
17         //Do nothing if you are out of health
18         if (health <= 0)
19             return;
20         //play the sound effect
21         HurtSoundEffect.Play();
22         //reduce the health
23         health -= value;
24         //Refresh the UI fillBar
25         fillBar.fillAmount = health / 100;
26         //Check if your health is zero or less => Dead
27         if (health <= 0)
28         {
29             FindObjectOfType<PlayerLife>().Die();
30         }
31     }
32

```

Figure 4.2.1.82: Script of Health Bar

Figure 4.2.1.83 shows the script of item collect and it is used to set the score of coins, trigger an audio when a player character collides with the coins, and show the score in the game. Full script was shown in appendix 23.

```

13 private void OnTriggerEnter2D(Collider2D collision)
14 {
15     if (collision.gameObject.CompareTag("coin"))
16     {
17         collectionSoundEffect.Play();
18         Destroy(collision.gameObject);
19         COIN += 10;
20         CoinText.text = " " + COIN ;
21     }
22 }
23 }
24
25 public void Deducecoin()
26 {
27     COIN -= 10;
28     CoinText.text = " " + COIN ;
29 }
30 public void Increasecoin()
31 {
32     COIN += 10;
33     CoinText.text = " " + COIN ;
34 }
35 }
36

```

Figure 4.2.1.83: Script of item collect

E. Coins and Checkpoint

The coins shown in the figure 4.2.1.84 are located in the random places of the game application and the checkpoints were located at the end of the level. All of the assets in figure 4.2.1.84 have boxes collide and each of them have different scripts to carry out the function. For the coin, the score of the coin will increase when the player character collides with the coin and the coin will disappear by inserting the script “itemCollector”. The checkpoint also inserts a “Checkpoint” script to let the player pass to the next level when the player fulfills the requirement.



Figure 4.2.1.84: Coins, and Checkpoint.

Figure 4.2.1.85 shows the inspector of coins and it is used to set the box collider 2D to collider with the player character and animator to set the animation of the coins.

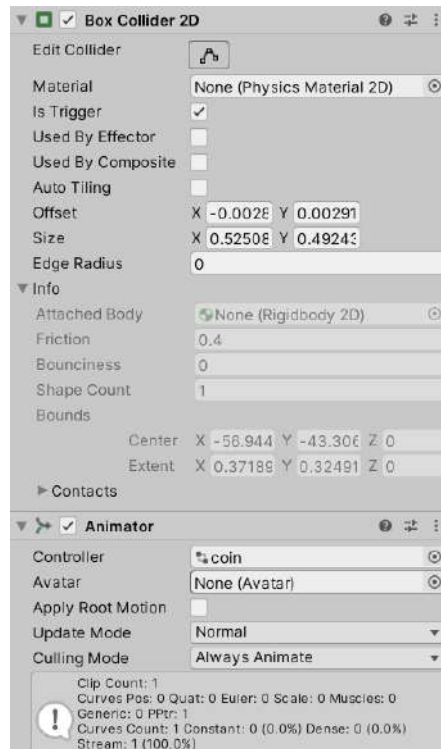


Figure 4.2.1.85: Inspector of coins

Figure 4.2.1.86 was used to set the function of the checkpoint. The box collider 2D was used to collide with player characters. The animator was used to set the animation of the checkpoint, trigger an audio when the player character collided with it and the checkpoint tutorial was used to set the requirement of passing the level.

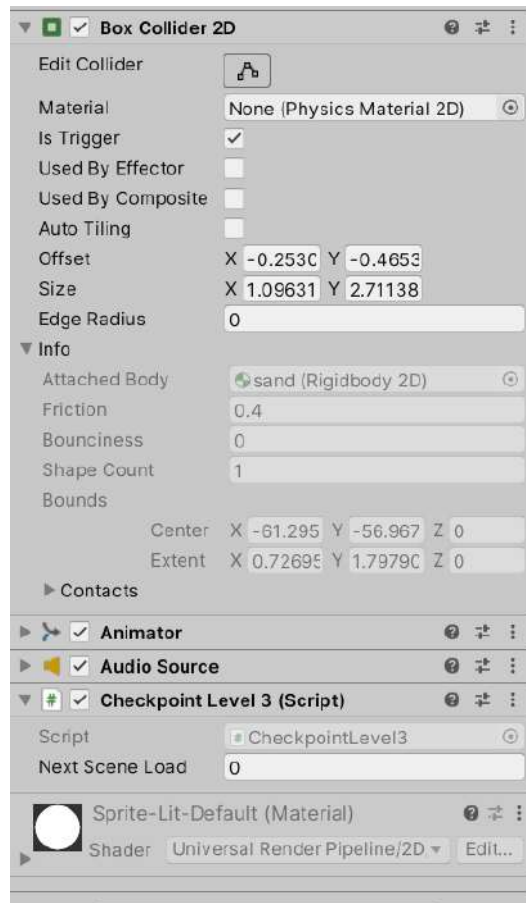


Figure 4.2.1.86: Inspector of checkpoint

Figure 4.2.1.87 shows the script of the checkpoint tutorial and it uses the requirement of the level. The script was set and the player character needed to fulfill the requirement to pass the level and the script will repeat for different with different requirements. A panel also will pop out when the player character did not fulfill the requirement. The loading screen will pop out when the player has fulfilled the requirement and passed to the next level. Full script was shown in appendix 24.

```

15 void Start()
16 {
17     nextSceneLoad = SceneManager.GetActiveScene().buildIndex + 1;
18     finishSound = GetComponent();
19     anim = GetComponent<Animator>();
20 }
21
22 private void OnTriggerEnter2D(Collider2D collision)
23 {
24     if (collision.gameObject.name == "Player" && !levelCompeted)
25     {
26         if (itemCollect.COIN >= 10)
27         {
28             anim.Play("win");
29             finishSound.Play();
30             levelCompeted = true;
31             FindObjectOfType<LoadingScreen2>().LoadScene();
32
33             if (SceneManager.GetActiveScene().buildIndex == 6)
34             {
35                 Debug.Log("You Win Game");
36                 FindObjectOfType<PlayerLife>().resetScore();
37             }
38             else
39             {
40                 Invoke("CompleteLevel", 2f);
41                 if (nextSceneLoad > PlayerPrefs.GetInt("levelAt"))
42                 {
43                     PlayerPrefs.SetInt("levelAt", nextSceneLoad);
44                     FindObjectOfType<PlayerLife>().resetScore();
45                 }
46             }
47         }
48         Debug.Log("Level " + PlayerPrefs.GetInt("levelAt") +
49
...ity\PSM\Assets\Script\Checkpoint\CheckpointTutorial.cs
"Unlocked");
50     }
51     else
52     {
53         levelCompeted = false;
54         FindObjectOfType<PlayerLife>().insufficientCoin();
55     }
56 }
57
58 }
59
60 private void CompleteLevel()
61 {
62     SceneManager.LoadScene(nextSceneLoad);
63     FindObjectOfType<PlayerLife>().resetScore();
64 }
65 }
66 }
67
68

```

Figure 4.2.1.87: Script of Checkpoint tutorial.

The figure 4.2.1.88 was the script of not enough coin panels and it is used to show the score of coins collected to let players know that they have fulfilled the requirement to pass the level. Full script was shown in appendix 25.

```

6 public class NotEnoughCoin : MonoBehaviour
7 {
8     [SerializeField] public Text CText;
9     // Start is called before the first frame update
10    void Start()
11    {
12        CText.text = " x " + itemCollect.COIN;
13    }
14 }
15 }
16

```

Figure 4.2.1.88: Script of Not Enough Coin

F. Camera

The figure 4.2.1.89 showed the inspector of the camera and it used to let the camera follow the player character.

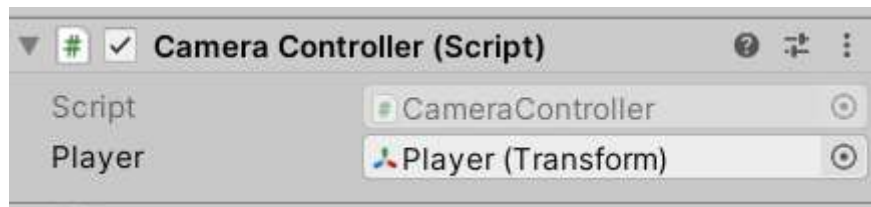


Figure 4.2.1.89: Inspector of Camera

The script in figure 4.2.1.90 was used to set the camera to follow the player character. Full script was shown in appendix 26.

```
5 public class CameraController : MonoBehaviour
6 {
7     [SerializeField] private Transform player;
8
9     // Update is called once per frame
10    private void Update()
11    {
12        transform.position = new Vector3(player.position.x, player.position.y,
13        transform.position.z);
14    }
15 }
```

Figure 4.2.1.90: Script of Camera Controller

F. NPC

The NPC in figure 4.2.1.91 will follow the player character when the player character is moving and the NPC will help to guide the player to play the game.



Figure 4.2.1.91: NPC

The Inspector in figure 4.2.1.92 was used to set the function of the NPC. It can be used to set the speed and the distance between player characters.

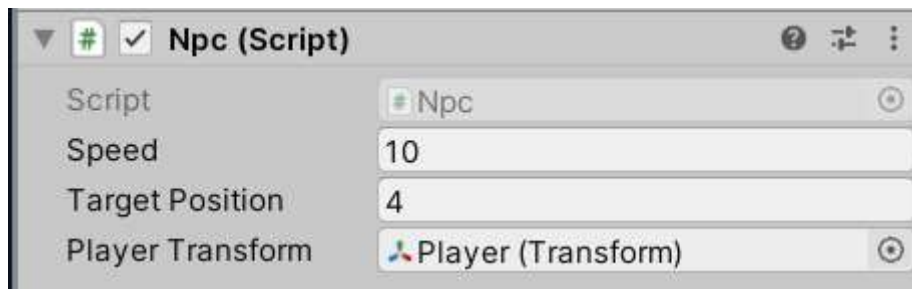


Figure 4.2.1.92: Inspector of NPC.

The script in figure 4.2.1.93 was used to let the NPC follow the player character. Full script shown in appendix 27.

```

20 void Update()
21 {
22     TargetFollow();
23     flipSprite();
24 }
25
26 void TargetFollow()
27 {
28     if (Vector2.Distance(transform.position, target.position) >
29         targetPosition)
30     {
31         transform.position = Vector2.MoveTowards(transform.position,
32             target.position, speed * Time.deltaTime);
33     }
34 }
35 void flipSprite()
36 {
37     if(playerTransform.position.x > transform.position.x)
38     {
39         //Face right
40         //transform.localScale = new Vector3(1,1,1) <- cannot use
41         //will resize the gameObject become bigger
42         transform.localScale = new Vector3(Mathf.Abs
43             (transform.localScale.x),transform.localScale.y,transf
44             le.z);
45     }
46     else if(playerTransform.position.x < transform.position.x)
47     {
48         //face left
49         ///transform.localScale = new Vector3(-1,1,1) <- ca
50         //it will resize the gameObject become bigger
51         transform.localScale = new Vector3(-1* Mathf.Abs
52             (transform.localScale.x), transform.localScale.y,
53             transform.localScale.z);
54     }
55 }
56 }
57 }
58 }
59 }

```

Figure 4.2.1.93: Script of NPC

4.2.2 Demonstrate

A. Demonstrate Prototype

After the build process, the full game application was demonstrated to the supervisor and friends to gain some feedback. The tools to demonstrate with the supervisor is Unity and the code was demonstrated using Microsoft Visual Studio 2019.

B. Gain Feedback

After demonstrating the game application to supervisor and friends, the feedback from supervisor and friends were recorded for the next process which is the Refine process to fix or change the User interface or game mechanism.

4.2.3 Refine

The Refine process was the last process for the prototype process. This process was used to fix the error, change the asset, user interface, code or game mechanism following the feedback that got from supervisor and friends. If the game development has not completed, the prototype will be recycled until the game development is completed.

A. Update requirement of proposed game




The requirements that get from supervisor and friends will be recorded and updated. Then, the bugs and the game were changed by following the requirements that were recorded.



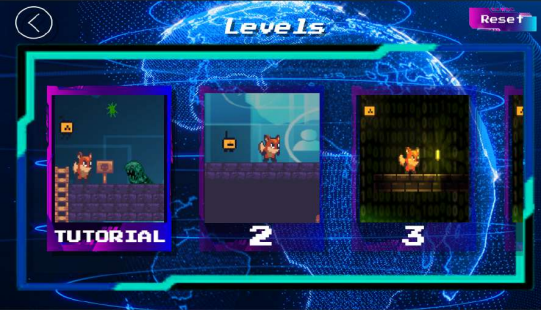

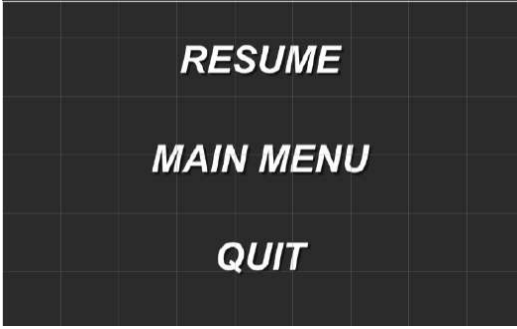
B. Fix the bug



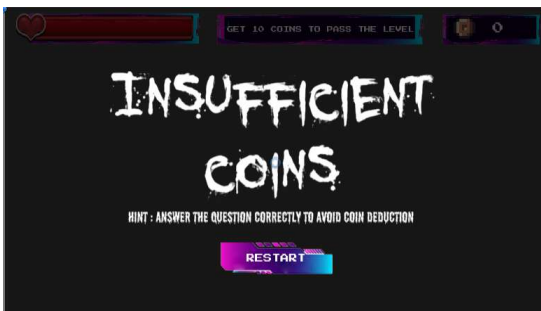
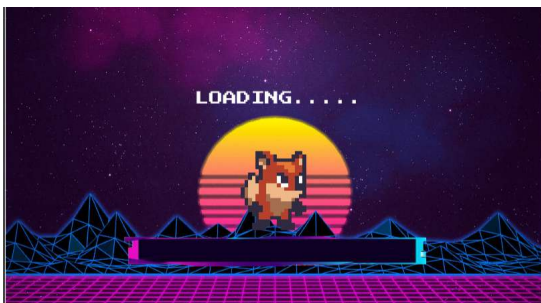
The bug will be fixed at this process and the requirement also will be updated in this process.

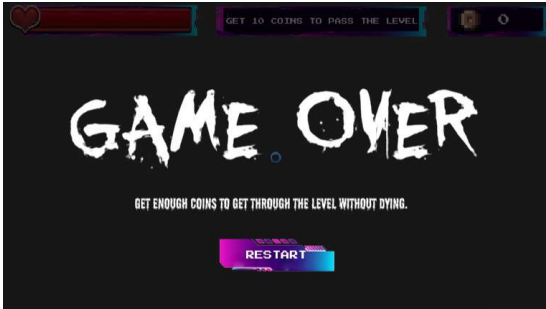
- Table update

Table 4.2.3.1: Requirement form

No	Preveiw version	Requirement	Final version
1.		<ol style="list-style-type: none"> 1. The User interface did not stick with the topic. 2. Add a confirm message before you quit the game. 	
2.		<ol style="list-style-type: none"> 1. The User interface of the game did not stick with the topic. 2. Add a sentence of requirement to pass the game. 3. Change the location of the score of coins collected. 	

<p>3.</p>		<p>1. Change the User Interface of the button.</p>	
<p>4.</p>		<p>1. Change the User Interface of the level page. 2. Let the level be locked when the player has not passed the level. 3. Change the back button symbol and location.</p>	
<p>5.</p>		<p>1. Change the User Interface of the Game Complete page.</p>	
<p>6.</p>		<p>1. Add a Restart button to let the player restart the level.</p>	

<p>7.</p>		<ol style="list-style-type: none"> 1. Change the User interface of the question. 2. Random the questions. 3. Make the color of the answer when it is correct or wrong. 	
<p>8.</p>		<ol style="list-style-type: none"> 1. Add insufficient coins panel 2. Set the button to restart the level. 3. Insert the Text Mesh Pro to the panel. 	
<p>9.</p>		<ol style="list-style-type: none"> 1. Add loading screen to solve the problem of loading longer time to another level. 2. Make animation of the player character. 3. Write the script for the loading process bar. 	

10.		<ol style="list-style-type: none"> 1. Add the game over the panel. 2. Add the function in the script of player life. 	
-----	--	--	---

- Functional acceptance testing

Functional acceptance test is used to uncover structural problems, hidden errors, and problems with specific components. The code of the proposed game needs to be updated or changed in a refined process after getting feedback from the demonstration process. The table was used to record all the tests that have been tried before and the actions that have been taken to solve the problem or errors. The things that going to test is test data, pre-condition is the things need to do before testing start, Post-condition is the result if the assets function well, test step is the step to test the asset, expected result is the result that is expected, actual result is the actual result after testing, and the action is the action that taken to solve the problem. The table shows in appendix 29.

4.3 Testing (UAT)

Result of User Acceptance Test (UAT)

The following results are collected from the User Acceptance Test and Feedback survey. The survey was carried out anonymously and there are a total 20 responses collected.

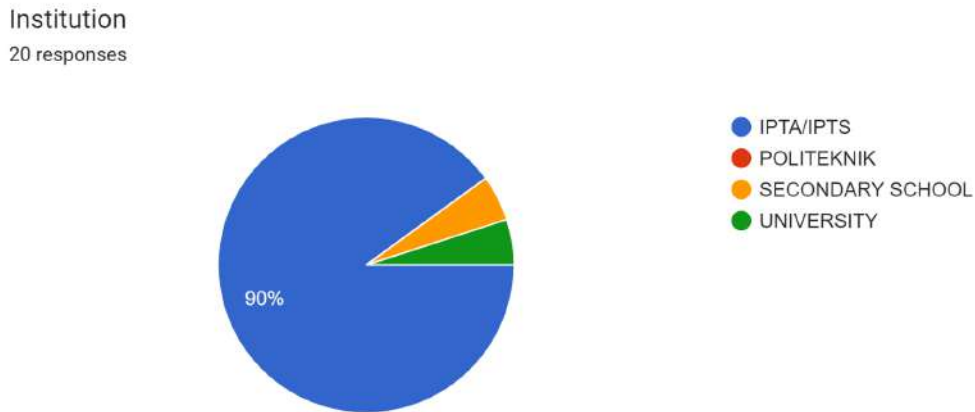


Figure 4.3.1: Result for the Institution.

The figure shown in figure 4.3.1 was the result of institutions of the 20 responses and 18 responses (90%) fall under the IPTA/IPTS, the 1 response was fall under secondary school and 1 response was fall under University.

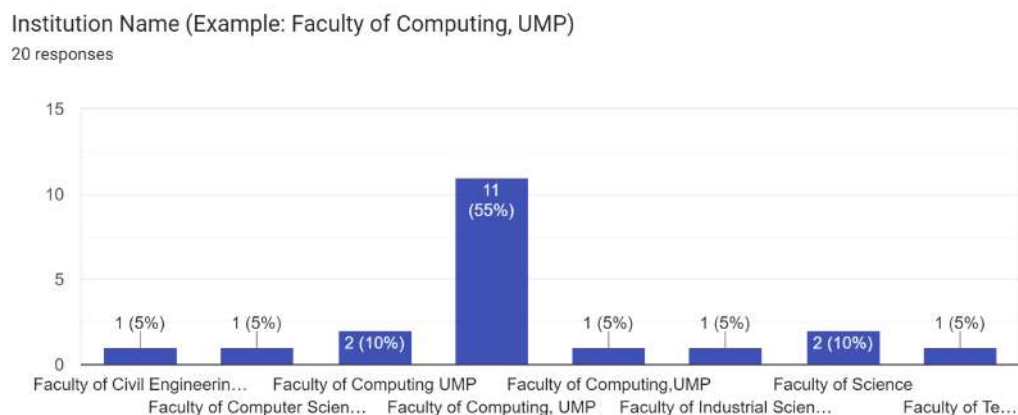


Figure 4.3.2: Result of Institution name.

Based on the result shown in figure 4.3.2 was the result of institution name. There are 1 response (5%) is Faculty of Civil Engineering Technology, UMP, 1 response (5%) is Faculty of Computer Science,UM, 2 responses(10%) are Faculty of Computing UMP, 11 responses(55%) are Faculty of Computing, UMP, 1 response(5%) is Faculty of Computing,UMP , 1 response (1%) is Faculty of Industrial Sciences and Technology, 2 responses (10%) are Faculty of Science and 1 response(5%) is Faculty of Technology Electrical and Electronic Computer System.

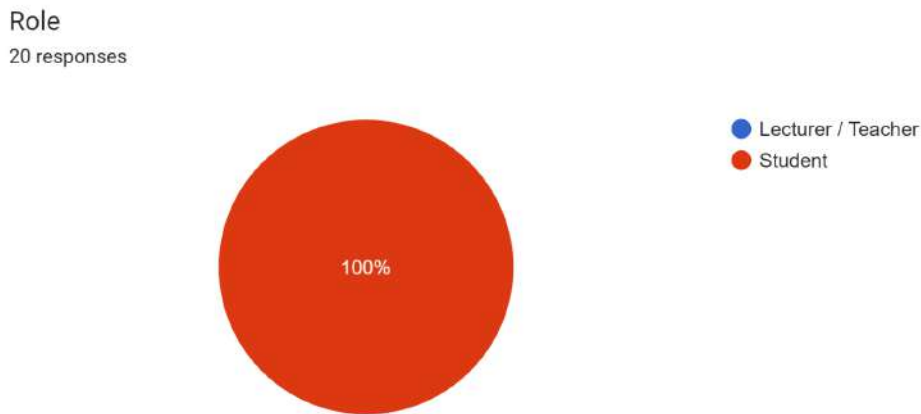


Figure 4.3.3: Result of Role of survey

The figure 4.3.3 showed the role of the survey. 20 responses(100%) are students.

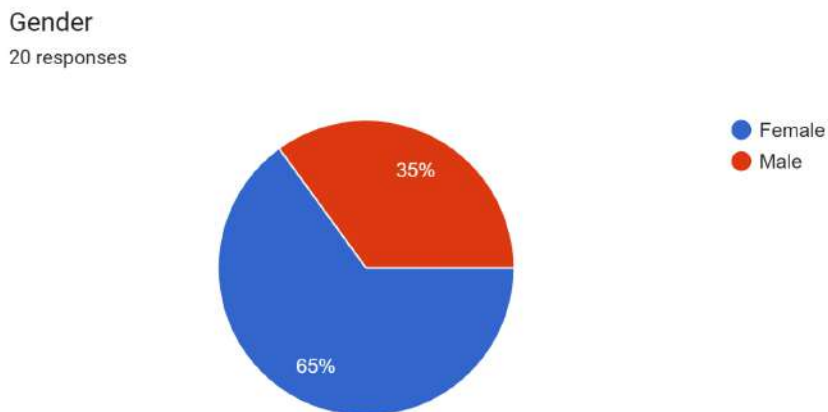


Figure 4.3.4: Result of gender.

The figure 4.3.4 shows the gender of the survey. There are 13 responses (65%) are female and 7 responses (35%) are male.

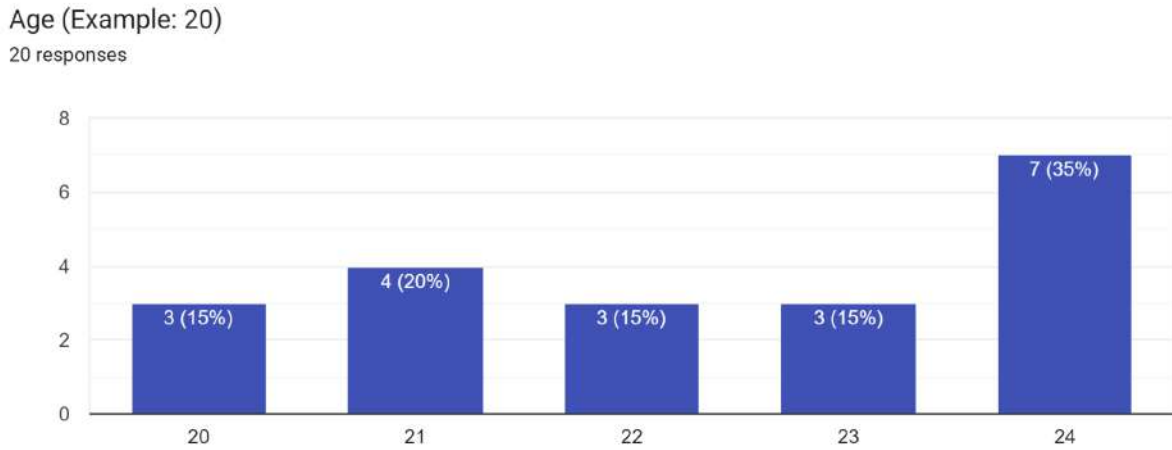


Figure 4.3.5: Result of age of survey.

The figure 4.3.5 shows the result of age. There are 3 responses (15%) are 20 years old. 4 responses (20%) are 21 years old, 3 responses (15%) are 22 years old, 3 responses (15%) are 23 years old and 7 responses (35%) are 24 years old.

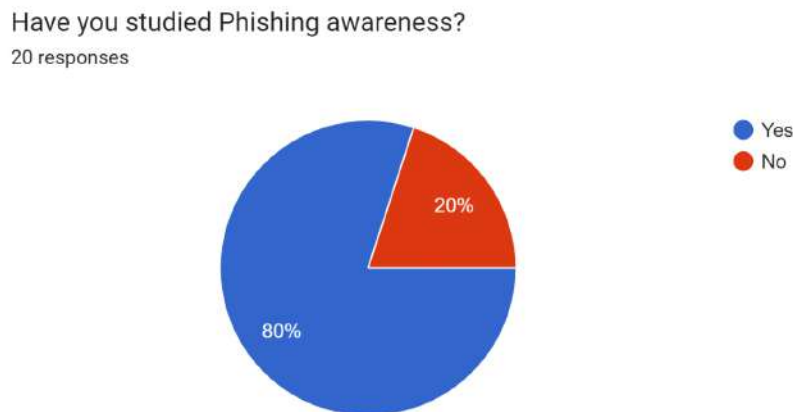


Figure 4.3.6: Result of Have You Studied Phishing Awareness.

The figure 4.3.6 showed the result of the survey for those who have studied phishing awareness. There are 16 responses (80%) who have studied phishing awareness and 4 responses who have not studied phishing awareness.

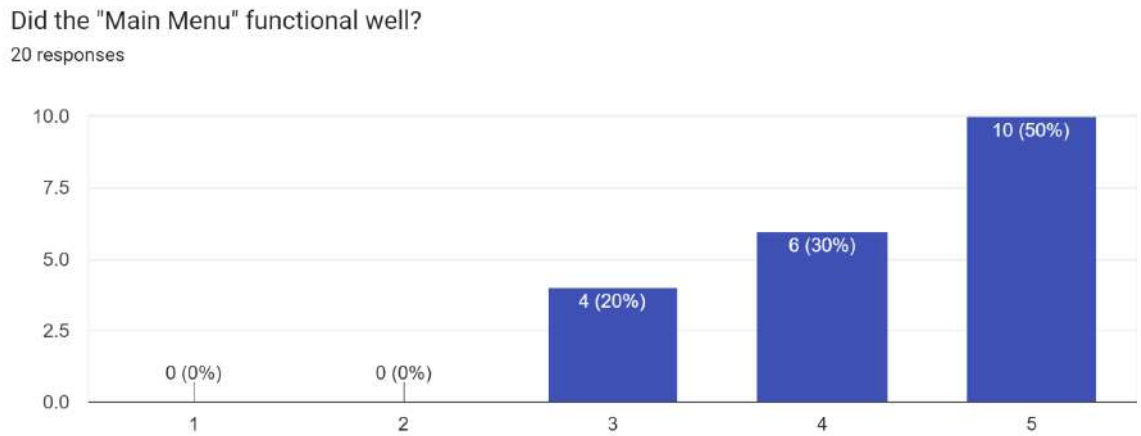


Figure 4.3.7: Result of the rate of main menu function

The figure 4.3.7 shows the result of the rate of the main menu function. There are 4 responses(20%) who agree with the function of the main menu and rated it 3 out of 5, 6 responses(30%) rate the function of the main menu function 4 out of 5 and 10 responses(50%) strongly agree and rated it 5 out of 5. Most of the responses agree that the main menu functions well.

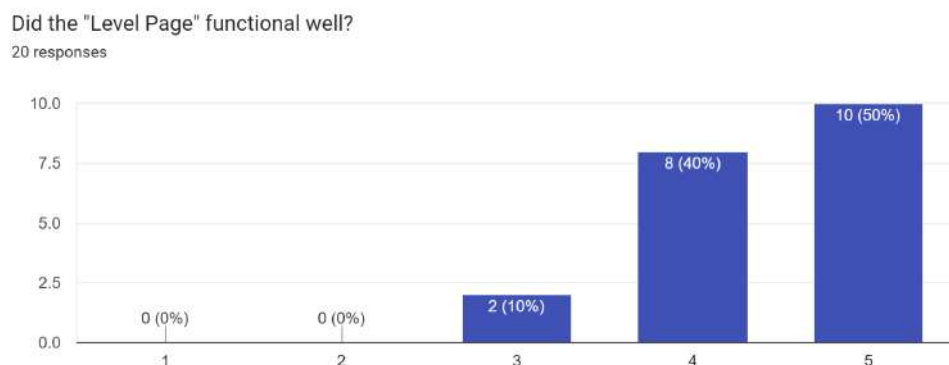


Figure 4.3.8: Rate of the Level page function.

The figure 4.3.8 shows the rate of the level page function and there are 2 responses(10%) who agree with the function of the level page and rated it as 3 out of 5. There are 8 responses(40%) who rated the function of level page 4 out of 5, and 10 responses (50%) strongly agree and rated it 5 out of 5. Most of the responses agree that the level page functions well.

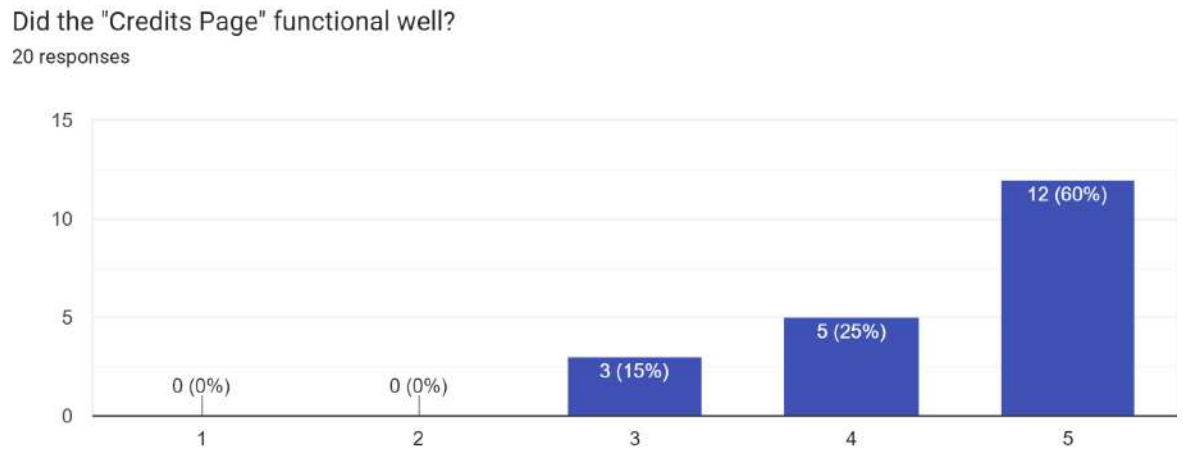


Figure 4.3.9: Rate of function of the Credits page.

The figure 4.3.9 shows the rate of the function of the credits page and there are 3 responses (15%) agree to the function and rate it as 3 out of 5. There are 5 responses (25%) who rated the function of the credits page 4 out of 5 and 12 responses (60%) strongly agree and rated it 5 out of 5. Most of the responses agree that the credits page functions well.

Did the "Question Page" functional well?

20 responses

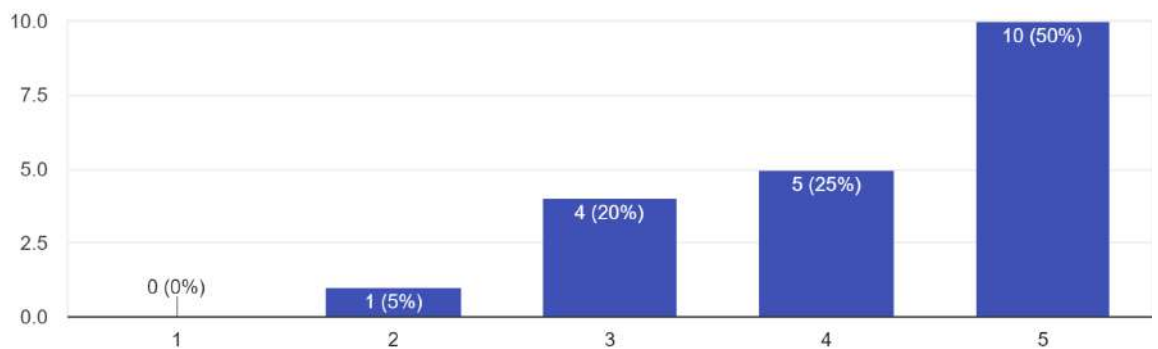


Figure 4.3.10: Rate of the function of the Question page.

There are 1 response (5%) rated the function of the question page 2 out of 5 as shown in the figure 4.3.10. There are also 4 responses (20%) who agreed and rated it 3 out of 5 and 5 responses (25%) rated it 4 out of 5. There also 10 responses (50%) strongly agree and rated it as 5 out of 5. Most of the responses agree that the question page functions well.

Did the player character in the game function well?

20 responses

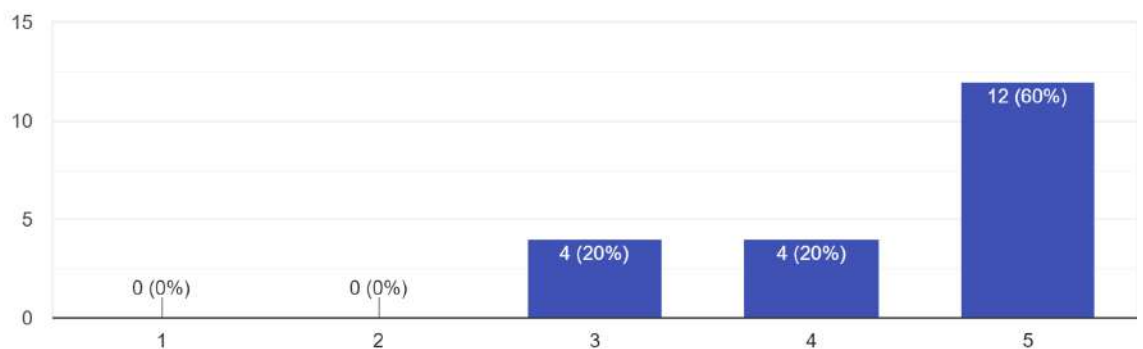


Figure 4.3.11: Rate of the function of player character

There are 4 responses (20%) rated it as 3 out of 5 and 4 responses (20%) rated it as 4 out of 5 as shown in the figure 4.3.11. There also 12 responses (60%) strongly agree and

rated it as 5 out of 5. Most of the responses agree that the function of the player character functions well.

Did the NPC in game follow the player character and chat with the player?
20 responses

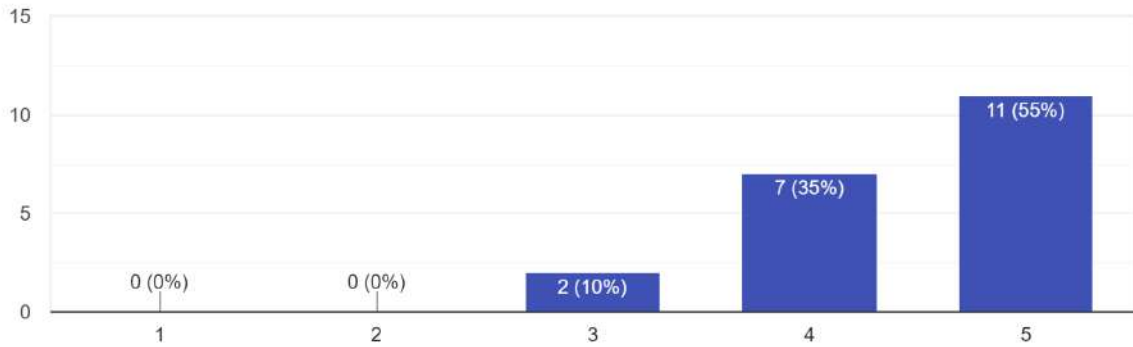


Figure 4.3.12: Rate of the function of the NPC.

The figure shows the rate of the function of the NPC and there are 2 responses (10%) rated it as 3 out of 5 and 7 responses (35%) rated it as 4 out of 5. There are also 11 responses (55%) who strongly agree with the function and rated it as 5 out of 5. Most of the responses agree that the NPC functions well.

Did the Question panel pop out and function well when the player character collides with the computer in the game?
20 responses

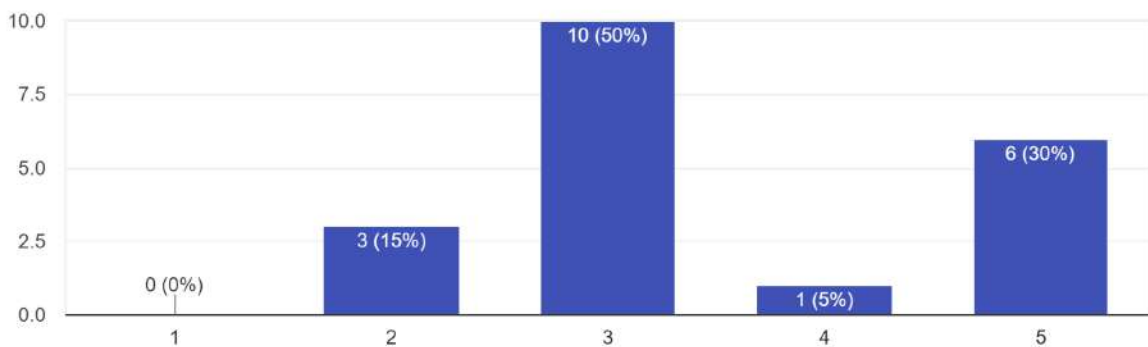


Figure 4.3.13: Rate of the function of the question panel.

From figure 4.3.12, there are 10 responses (50%) are rated it as 3 out of 5 and 3 responses (15%) are rated it as 3 out of 5. There also has 1 response (5%) rated as 4 out of 5 and 6 responses (30%) rated as 5 out of 5. Most of the responses agree that the question panel functions well. However, there still needs some improvement in the question panel because most of the responses just rated it as 3 out of 5.

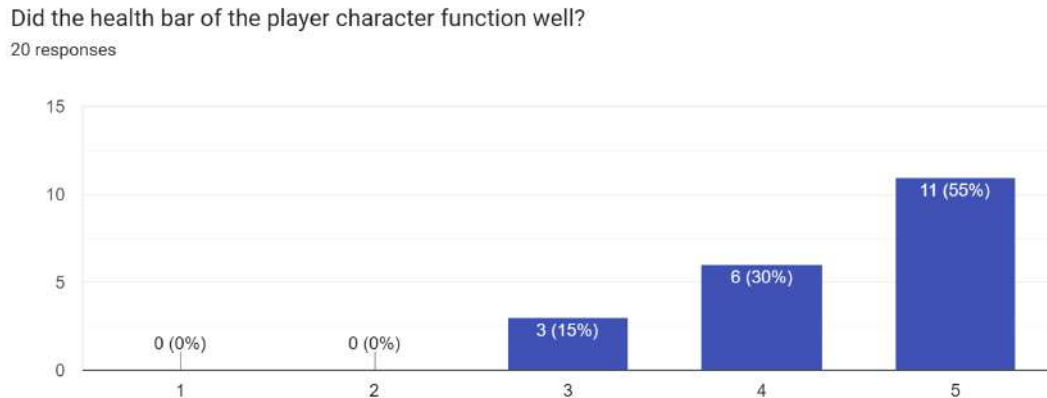


Figure 4.3.14: Rate of the function of the health bar of the player character.

The figure 4.3.14 showed the rate of the function os health bar of player character and there are 3 responses (15%) rated it as 3 out of 5. There are 6 responses (30%) are rated as 4 out of 5 and 11 responses (55%) strongly agree and rated as 5 out of 5. Most of the responses agree that the health bar of the player character functions well.

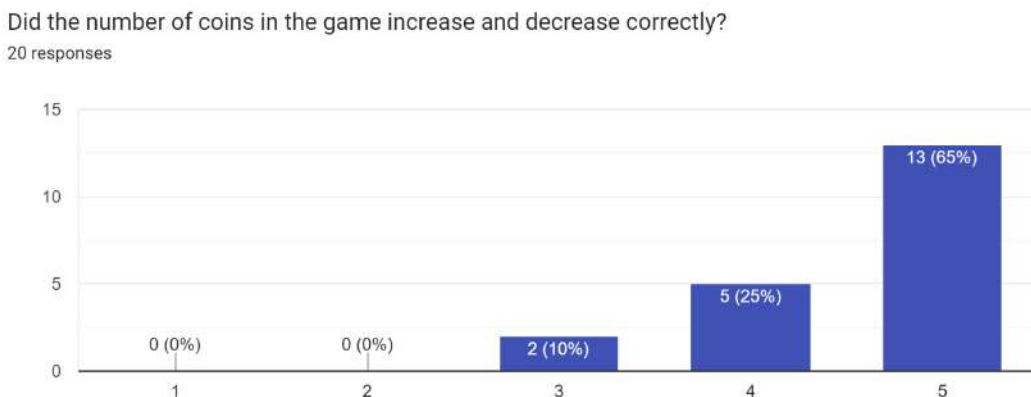


Figure 4.3.15: Rate of function of coin.

The figure 4.3.15 showed most of the responses which are 13 responses (65%) strongly agree and rated as 5 out of 5. There also 5 responses (25%) rated it as 4 out of 5 and 2 responses (10%) rated it as 3 out of 5. Most of the responses agree that the question page functions well.

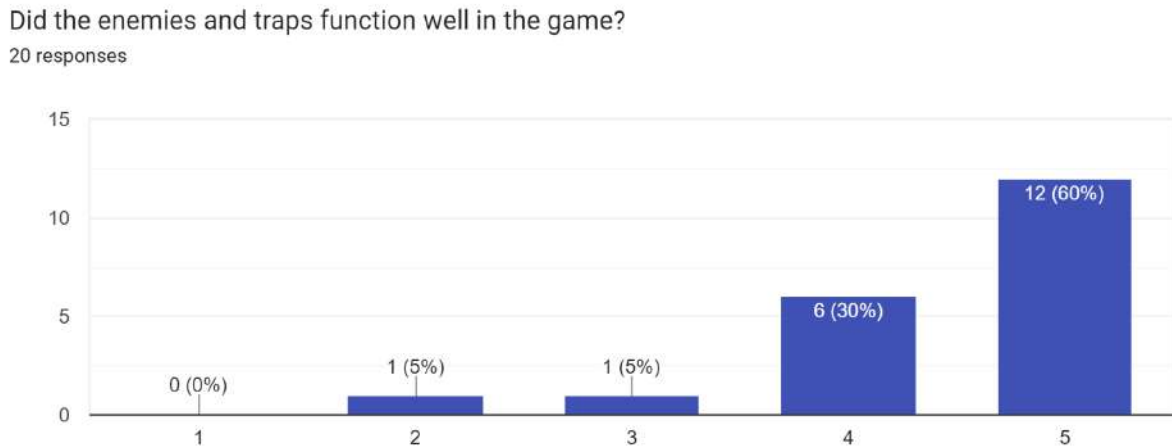


Figure 4.3.16: Rate of function of the enemies and traps.

There are 12 responses (60%) strongly agree and rated the function of the enemies and traps 5 out of 5, 6 responses (30%) rated 4 out of 5, 1 response (5%) rated it as 3 out of 5 and 1 response (5%) rated it as 2 out of 5 and result was shown in figure 4.3.16. Most of the responses agree that the enemies and traps in the game function well.

Did the light and neon system function well in the game?

20 responses

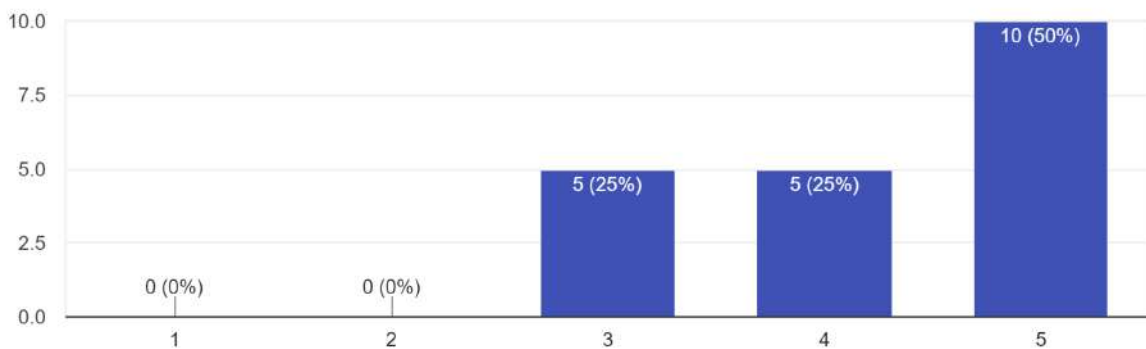


Figure 4.3.17: Rate of function of light and neon system.

Most of the 10 responses (50%) strongly agree that the light and neon system function well in the game and rated it as 5 out of 5, 5 responses (25%) rated it as 4 out of 5, 5 responses (25%) rated it as 3 out of 5 and the result is shown in the figure 4.3.17. Most of the responses agree that the light and neon system functions well.

Did the player character stick on the moving platform after the player jumped on?

20 responses

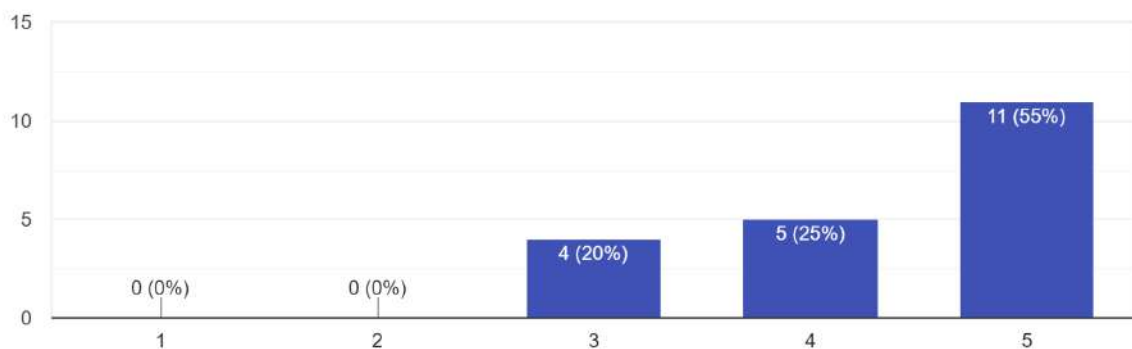


Figure 4.3.18:Rate of the function of the moving platform.

Most of the 11 responses (55%) strongly agree the moving platform is functioning well and rated it as 5 out of 5, 5 responses (25%) rated it as 4 out of 5 and 4 responses (20%)

rated it as 3 out of 5 and the result of the rate was shown in the figure 4.3.18. Most of the responses agree that the moving platform functions well.

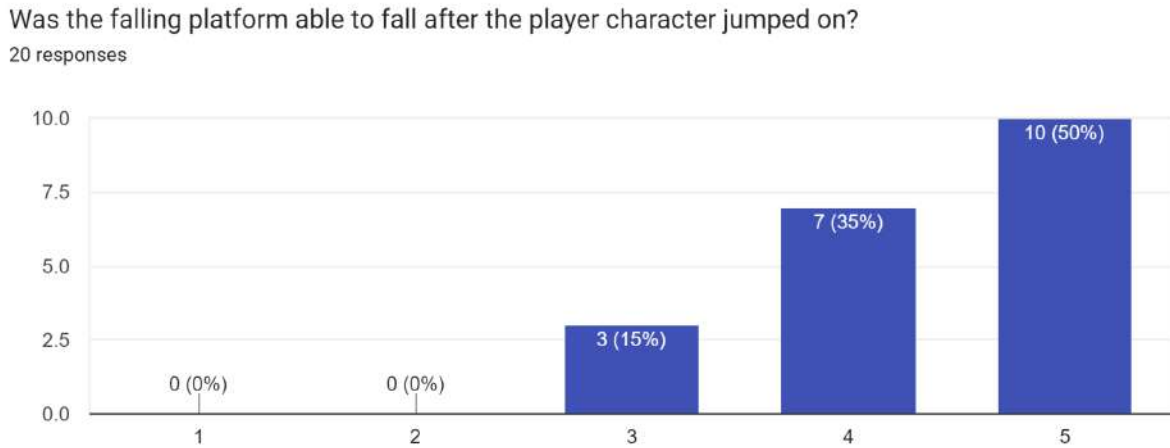


Figure 4.3.19: Rate of the function of the falling platform.

Most of the 10 responses (50%) strongly agree that the falling platform functions well and rated it as 5 out of 5 as shown in the figure 4.3.19. 7 responses (35%) rated it as 4 out of 5 and 3 responses (15%) rated it as 3 out of 5. Most of the responses agree that the falling platform functions well.

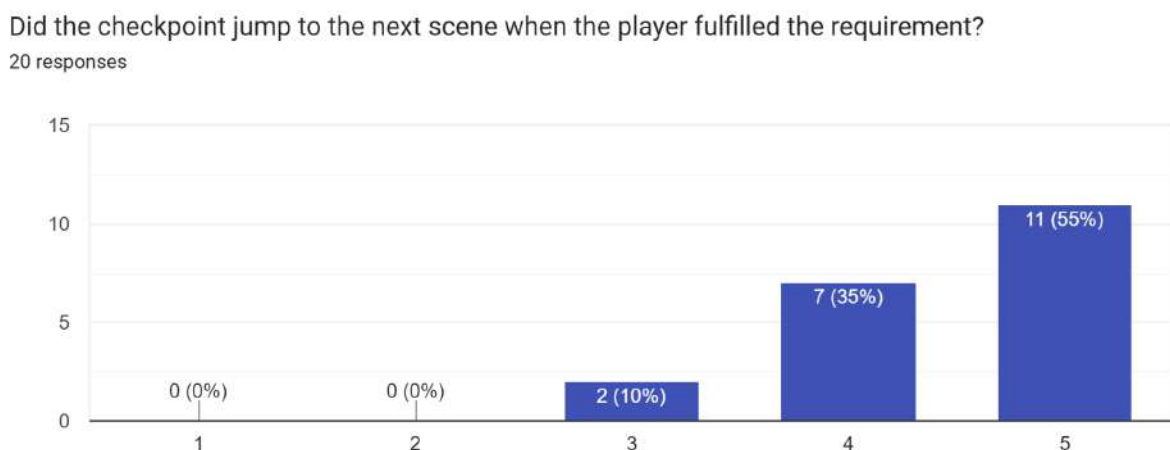


Figure 4.3.20: Rate of the function of the checkpoint.

The figure 4.3.20 showed the result of the rate of the function of the checkpoint and most of the 11 responses (65%) strongly agreed the checkpoint was functioning well and rated it as 5 out of 5. There are 7 responses (35%) rated it as 4 out of 5 and 2 responses (10%) rated it as 3 out of 5. So, most of the responses agree the checkpoint functions well to pop out the correct panel or jump to the next level when the player character collides with it.

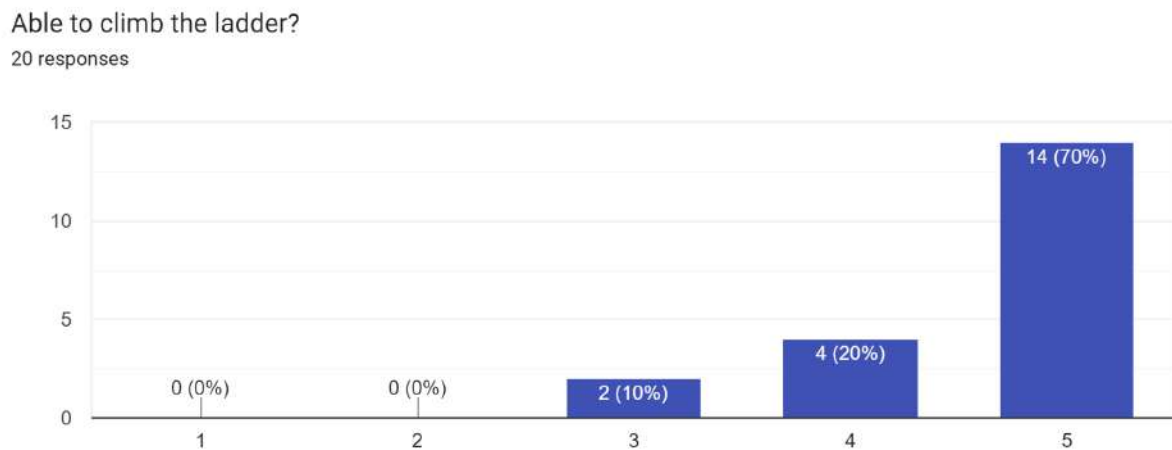


Figure 4.3.21: Rate of function of ladder.

The figure 4.3.21 showed the result of the rate of the function of the ladder and most of the 14 responses (70%) strongly agree that the function of the ladder was functioning well and rated it as 5 out of 5. There also 4 responses (20%) rated it as 4 out of 5 and 2 responses (10%) rated it as 3 out of 5. Most of the responses strongly agree the ladder was functioning well.

What is the error and problem that is still in the game?

8 responses



Figure 4.3.22: Comment of the function of the game.

There are some of the comments that were written by the responses and there still some improvement on the function and loading speed of the game as shown in the figure 4.3.22.

Do you like the Smell Phishy game?

20 responses

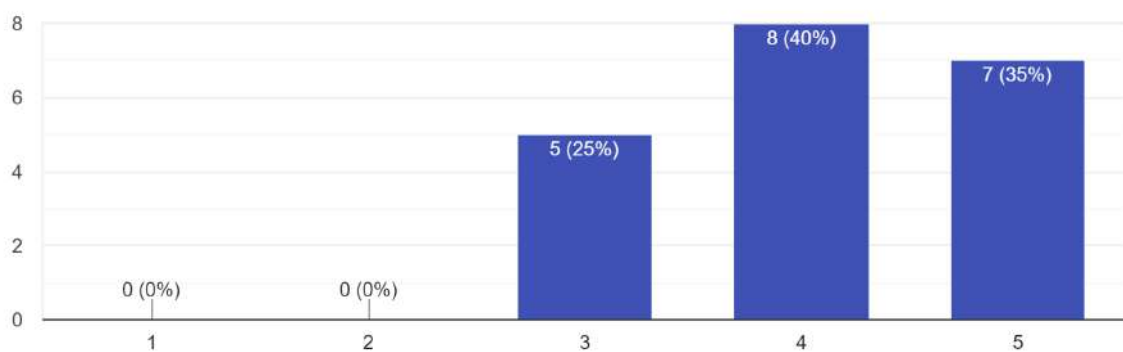


Figure 4.3.23: Rate of the likes of the game.

The figure 4.3.23 showed most of the 8 responses (40%) rated it as 4 out of 5, 7 responses (35%) rated it as 5 out of 5 and 5 responses (25%) rated it as 3 out of 5. So, the

game needs some improvement such as adding more game mechanisms to make the game more interesting and attractive.

Do you enjoy playing the Smell Phishy game?

20 responses

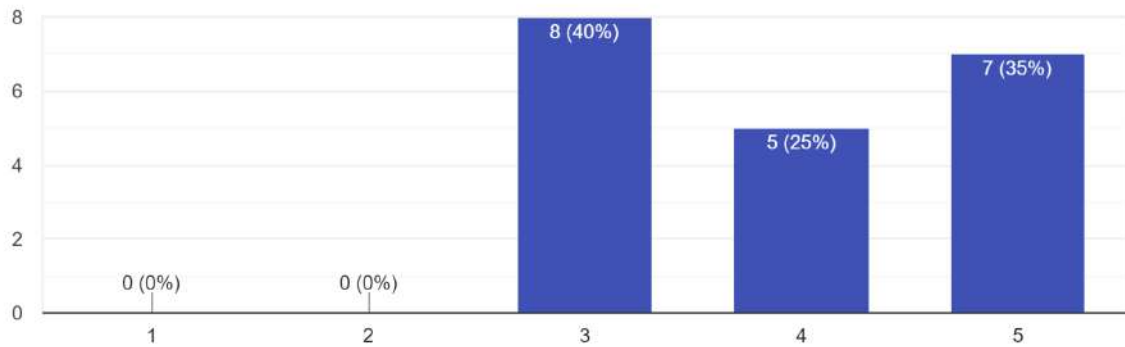


Figure 4.3.24: Rate of game enjoyment.

Most of the 8 responses (40%) rated the game enjoyment as 3 out of 5, 5 responses (25%) rated it as 4 out 5 and 7 responses (35%) rated it as 5 out of 5 as shown in the figure 4.3.24. Most of the responses enjoy the game but the game still needs to improve.

Did you gain information when playing the game?

20 responses

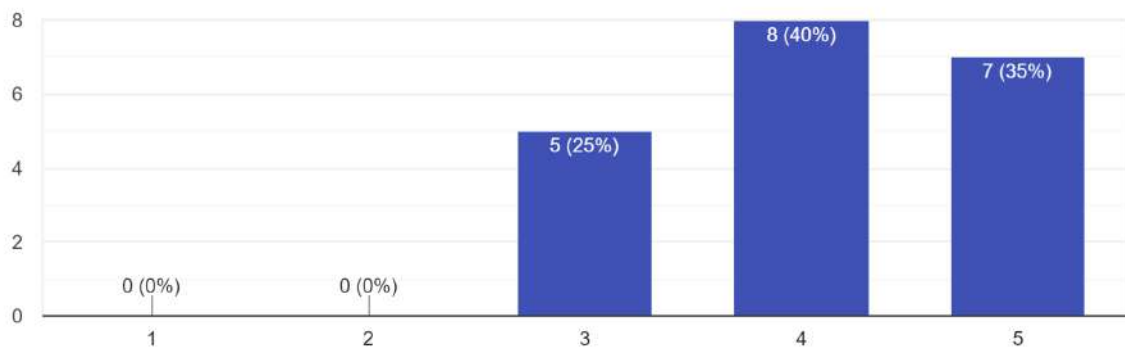


Figure 4.3.25: Rate of did player gain knowledge from the game.

The figure 4.3.25 showed most of the 8 responses (40%) rated it as 4 out of 5, 7 responses (35%) rated it as 5 out of 5 and 5 responses (25%) rated it as 3 out of 5. Most of the responses are gaining knowledge after testing the knowledge after playing the game, but the game still needs some improvement.

Do you want to play the Smell again game again?

20 responses

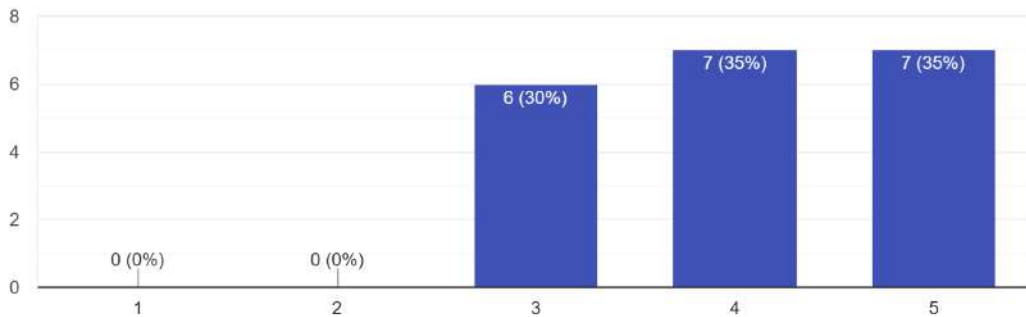


Figure 4.3.26: Rate of Do players want to play the game again.

There are 7 responses (35%) rated 5 out of 5 and 7 responses (35%) rated 4 out of 5. There were also 6 responses (30%) rated 3 out of 5 and the result was shown in the figure 4.3.26. Most of the responses will play the game again.

Do you think the Smell Phishy can help educate on Phishing attack awareness?

20 responses

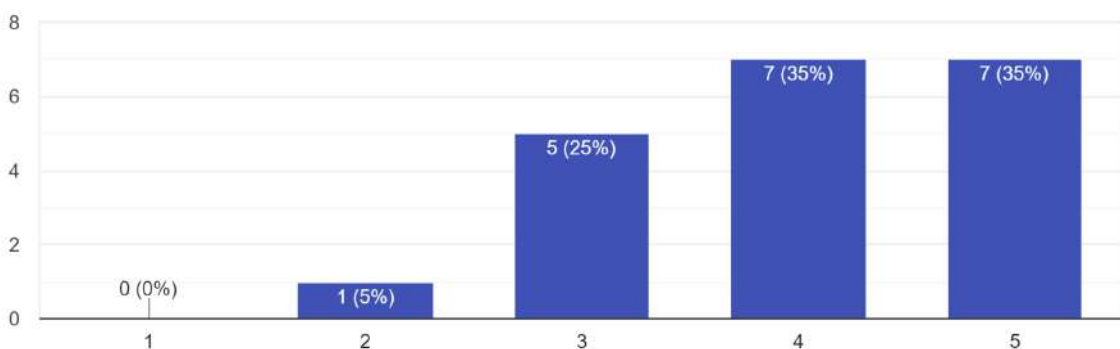


Figure 4.3.27: Rate of marketability of the game.

The figure 4.3.27 showed most of the 7 responses (35%) rated 4 out of 5 and the same number of the responses rated 5 out of 5. There were also 5 responses (25%) rated 3 out of 5 and 1 response (5%) rated 2 out of 5. So, the marketability of the game may be an interesting way to test the player about phishing awareness but that still needs some improvement.

What is your comment(s) on the game?

7 responses



Figure 4.3.28: Comments of the game.

There are some comments from the responses that showed in the figure 4.3.28. There are some suggestions to improve the game. The game can be more game mechanism as the suggestion showed in the figure 4.3.28.

4.4 Implementation

This was the last process before publishing the game application. This process was very important and this process was going to do the system simulation, issue tracking, debugging and final update. After all of the issues or bugs have been fixed, the game will be exported as a game application and uploaded to itch.io.

CHAPTER 5

CONCLUSION

5.1 Objective Revisited

In this project, we expected users could get benefits after playing “Smell Phishy”, the 2D platform game that developed using Unity. Players need to control the player character to pass the level by avoiding colliding with the obstacles. The players can answer the question in the game to know more about phishing awareness and test themselves. By the gameplay of the game, players can learn or test the knowledge of their Phishing awareness that they learned and used the knowledge in the real world to prevent becoming a victim of a phishing attack.

This game is only available on PC platform and players need to download and install from itch.io. The game is easy to play and understand and players can gain knowledge and test themselves in a fun way.

5.2 Limitation

During the development of the game, there are some problems that have been faced. The first problem was the knowledge of game development. Lack of knowledge of game development such as coding of the game mechanisms, animation, user interface and so on. The developer needs to learn some new skills, design, develop and test the game to maintain the quality of the game. This is because not all the skills can be learned from the class and need motivation to learn new skills to make the game more attractive. However, lack of knowledge limits developers from even having many interesting ideas.

Besides that, coding skills were one of other problems faced by developers. The developer needs to learn some new coding for new game mechanics and it is difficult to solve it when some problem happens without any errors.

Lack of knowledge of level design also became one of the limitations of the developer. The developer will design the level more difficult when the level increases.

However, the difficulty of the level needs to be increased by the different level design. So, developers need to design on the level, but lack of knowledge of level design will make the difficulty of the level the same.

5.3 Future Work

For future work, this game will update based on the comments that received from the player and solve the problem that faced by the player. The question also will be updated and increased to test the player more about the phishing awareness to help them increase their knowledge. The game can develop more game mechanics to make the game more interesting.

REFERENCES

1. Phishing Awareness. (2022, March 2). LeTourneau University - Knowledge Base. <https://servicedesk.letu.edu/TDClient/2460/Portal/KB/ArticleDet?ID=123658>
2. Daengsi, T. (2021, November 15). *Cybersecurity Awareness Enhancement: A Study of the Effects of Age and Gender of Thai Employees Associated with Phishing Attacks*. SpringerLink. https://link.springer.com/article/10.1007/s10639-021-10806-7?error=cookies_not_supported&code=f5606e4c-6050-48eb-b5a8-b917acf9690f
3. Staff, S. (2022, November 7). *4 recommendations to combat phishing*. Security Magazine.
4. <https://www.securitymagazine.com/articles/98585-4-recommendations-to-combat-phishing>
5. Dhamija, R., Tygar, J. D., & Hearst, M. (2006). Why phishing works. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. <https://doi.org/10.1145/1124772.1124861>
6. Jansson, K., & von Solms, R. (2013). Phishing for phishing awareness. *Behaviour & Information Technology*, 32(6), 584–593. <https://doi.org/10.1080/0144929x.2011.632650>
7. CJ, G., Pandit, S., Vaddepalli, S., Tupsamudre, H., Banahatti, V., & Lodha, S. (2018). PHISHY - A Serious Game to Train Enterprise Users on Phishing Awareness. *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*. <https://doi.org/10.1145/3270316.3273042>

8. Hale, M. L., Gamble, R. F., & Gamble, P. (2015). CyberPhishing: A Game-Based Platform for Phishing Awareness Testing. *2015 48th Hawaii International Conference on System Sciences*. <https://doi.org/10.1109/hicss.2015.670>
9. Arachchilage, N. A. G., & Love, S. (2013). A game design framework for avoiding phishing attacks. *Computers in Human Behavior*, 29(3), 706–714. <https://doi.org/10.1016/j.chb.2012.12.018>
10. Arachchilage, N. A. G., & Love, S. (2014). Security awareness of computer users: A phishing threat avoidance perspective. *Computers in Human Behavior*, 38, 304–312. <https://doi.org/10.1016/j.chb.2014.05.046>
11. Reinheimer, B., Aldag, L., Mayer, P., Mossano, M., Duezguen, R., Lofthouse, B., Von Landesberger, T., & Volkamer, M. (2020). An investigation of phishing awareness and education over time: When and how to best remind users. *Symposium on Usable Privacy and Security*, 259–284. https://www.usenix.org/system/files/soups2020-reinheimer_0.pdf
12. Weanquoi, P., Johnson, J., & Zhang, J. (2018). Using a Game to Improve Phishing Awareness. *Journal of Cybersecurity Education, Research and Practice*, 2018(2), 2.
13. Volkamer, M., Renaud, K., Reinheimer, B., Rack, P., Ghiglieri, M., Mayer, P., Kunz, A., & Gerber, N. (2018). Developing and Evaluating a Five Minute Phishing Awareness Video. *Trust, Privacy and Security in Digital Business*, 119–134. https://doi.org/10.1007/978-3-319-98385-1_9
14. Caputo, D. D., Pfleeger, S. L., Freeman, J. D., & Johnson, M. E. (2014). Going Spear Phishing: Exploring Embedded Training and Awareness. *IEEE Security & Privacy*, 12(1), 28–38. <https://doi.org/10.1109/msp.2013.106>
15. Downs, J. S., Holbrook, M. B., & Cranor, L. F. (2006). Decision strategies and susceptibility to phishing. *Proceedings of the Second Symposium on Usable Privacy and Security - SOUPS '06*. <https://doi.org/10.1145/1143120.1143131>

APPENDIXES

Appendix 1: User Acceptance Testing Form

Section 1: Test the functionality of the game.

Table 3.14: UAT about the functionality of the game.

NO	Event	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1.	Did the “Main Menu” function well?					
2.	Did the “Level Page” function well?					
3.	Did the “Credits Page” function well?					
4.	Did the “Question Page” in the game function well?					
5.	Did the player character in the game function well?					
6.	Did the NPC in game follow the player character and chat with the player?					

7.	Did the Question panel pop out and function well when the player character collides with the computer in the game?					
8.	Did the health bar of the player character function well?					
9.	Did the number of coins in the game increase and decrease correctly?					
10.	Did the enemies and traps function well in the game?					
11.	Did the light and neon system function well in the game?					
12.	Did the player character stick on the moving platform after the player jumped on it?					

13.	Was the falling platform able to fall after the player character jumped on it?					
14.	Did the checkpoint jump to the next scene when the player fulfilled the requirement?					
15.	Able to climb the ladder?					
14.	What is the error and problem that is still in the game?					

Section 2 - Test the usability and effectiveness of the game

Table 3.15: UAT about usability and effectiveness of the game

No	Questions	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1.	Do you like the Smell Phishy game?					
2.	Do you enjoy playing the Smell Phishy game?					

3.	Did you gain information when playing the game?					
4.	Do you want to play the Smell Phishy game again?					
5.	Do you want to play the Smell Phishy again?					
6.	Do you think the Smell Phishy can help educate on Phishing attack awareness?					
7.	What is your comment(s) on the game?					

Appendix 2: Script of Main Menu

```
...roject_unity\PSM\Assets\Script\User Interface\MainMenu.cs 1
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class MainMenu : MonoBehaviour
7 {
8     [SerializeField] public AudioSource buttonSource;
9     // Start is called before the first frame update
10    void Start()
11    {
12
13    }
14
15    // Update is called once per frame
16    void Update()
17    {
18
19    }
20    public void exit()
21    {
22
23        Application.Quit();
24        Debug.Log("Quit");
25    }
26    public void Tutorial()
27    {
28
29        SceneManager.LoadScene("Tutorial level 1");
30        FindObjectOfType<PlayerLife>().resetScore();
31    }
32
33    public void Level2()
34    {
35
36        SceneManager.LoadScene("Tutorial level 2");
37        FindObjectOfType<PlayerLife>().resetScore();
38    }
39
40    public void Level3()
41    {
42
43        SceneManager.LoadScene("Tutorial level 3");
44        FindObjectOfType<PlayerLife>().resetScore();
45    }
46
47    public void Level4()
48    {
49
```

```
50     SceneManager.LoadScene("Tutorial level 4");
51     FindObjectOfType<PlayerLife>().resetScore();
52 }
53
54 public void Level5()
55 {
56
57     SceneManager.LoadScene("Tutorial level 5");
58     FindObjectOfType<PlayerLife>().resetScore();
59 }
60
61 public void Level6()
62 {
63
64     SceneManager.LoadScene("Tutorial level 6");
65     FindObjectOfType<PlayerLife>().resetScore();
66 }
67
68 public void MainMenuPage()
69 {
70
71     SceneManager.LoadScene("Main Menu");
72     FindObjectOfType<PlayerLife>().resetScore();
73 }
74 }
75 }
76
```


Appendix 3: Script of Level Selection

```
..._unity\PSM\Assets\Script\User Interface\LevelSelection.cs 1
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class LevelSelection : MonoBehaviour
7 {
8     public int levelAt;
9     public int max_level;
10    public Button[] lvlButtons;
11
12    // Start is called before the first frame update
13    void Start()
14    {
15        levelAt = 1;
16        levelAt = PlayerPrefs.GetInt("levelAt", 1);
17
18    }
19
20    private void Update()
21    {
22        for (int i = 0; i < lvlButtons.Length; i++)
23        {
24            if (i + 1 > levelAt)
25            {
26                lvlButtons[i].interactable = false;
27                Debug.Log("" + levelAt);
28            }
29            else
30            {
31                lvlButtons[i].interactable = true;
32                Debug.Log("" + levelAt);
33            }
34        }
35    }
36
37    public void Reset()
38    {
39        levelAt = 1;
40        PlayerPrefs.SetInt("levelAt", levelAt);
41    }
42
43 }
44
```

Appendix 4: Script of Pause Menu

```
D:\Users\Acer\project_unity\PSM\Assets\Script\PauseMenu.cs 1
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class PauseMenu : MonoBehaviour
7 {
8     public static bool GameIsPaused = false;
9
10    public GameObject pauseMenuUI;
11
12    // Update is called once per frame
13    void Update()
14    {
15        if (Input.GetKeyDown(KeyCode.Escape))
16        {
17            if (GameIsPaused)
18            {
19                Resume();
20            }
21            else
22            {
23                Pause();
24            }
25        }
26    }
27    public void Resume()
28    {
29        pauseMenuUI.SetActive(false);
30        Time.timeScale = 1f;
31        GameIsPaused = false;
32    }
33
34    void Pause()
35    {
36        pauseMenuUI.SetActive(true);
37        Time.timeScale = 0f;
38        GameIsPaused = true;
39    }
40    public void MainMenu()
41    {
42        Time.timeScale = 1f;
43        SceneManager.LoadScene("Main Menu");
44    }
45    public void Quit()
46    {
47        Debug.Log("QUIT");
48        Application.Quit();
49    }
}
```

```
D:\Users\Acer\project_unity\PSM\Assets\Script\PauseMenu.cs 2
50    public void Restart()
51    {
52        SceneManager.LoadScene(SceneManager.GetActiveScene().name);
53        Time.timeScale = 1f;
54    }
55 }
56
```

Appendix 5: Script of Quiz answer and question

```
...ect_unity\PSM\Assets\Script\Quiz\QuizAnswerAndQuestion.cs 1
1
2
3 [System.Serializable]
4
5 public class QuizAnswerAndQuestion
6 {
7     public string Question;
8     public string[] Answer;
9     public int CorrectAnswer;
10
11 }
12
```

Appendix 6: Script of Answer

```
...Acer\project_unity\PSM\Assets\Script\Quiz\AnswerScript.cs 1
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class AnswerScript : MonoBehaviour
7 {
8     public static bool Paused = false;
9     public bool isCorrect = false;
10    public QuizManager quizManager;
11
12    public Color startColor;
13
14    private void Start()
15    {
16        startColor = GetComponent<Image>().color;
17    }
18    public void Answer()
19    {
20
21        if(isCorrect)
22        {
23            GetComponent<Image>().color = Color.green;
24            Debug.Log("correct Answer");
25
26            // put add coin and correct panel
27            Time.timeScale = 1f;
28            Paused = false;
29
30            quizManager.correct();
31        }
32        else
33        {
34            GetComponent<Image>().color = Color.red;
35            Debug.Log("Wrong Answer");
36
37            // put deduce coin and wrong panel
38            Time.timeScale = 1f;
39            Paused = false;
40
41            quizManager.wrong();
42        }
43    }
44 }
45
```

Appendix 7: Script of Quiz Manager

```
..er\project_unity\PSM\Assets\Script\ (D)Quiz\QuizManager.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using TMPro;
4 using UnityEngine;
5 using UnityEngine.UI;
6 public class QuizManager : MonoBehaviour
7 {
8     public List< QuizAnswerAndQuestion > QnA;
9     public GameObject[] options;
10    public int currentQuestion;
11    public GameObject lose;
12
13    public Text QuestionTxt;
14
15    int totalQuestion = 0;
16
17
18    private void Start()
19    {
20        totalQuestion = QnA.Count;
21        generateQuestion();
22    }
23
24    public void correct()
25    {
26        FindObjectOfType<itemCollect>().Increasecoin();
27        QnA.RemoveAt(currentQuestion);
28        FindObjectOfType<popOut>().PopExit();
29        StartCoroutine(WaitForNext());
30    }
31
32    public void wrong()
33    {
34        FindObjectOfType<itemCollect>().Deducecoin();
35        QnA.RemoveAt(currentQuestion);
36        FindObjectOfType<popOut>().PopExit();
37        StartCoroutine(WaitForNext());
38        //FindObjectOfType<popOut>().deduceLife();
39    }
40
41    IEnumerator WaitForNext()
42    {
43        yield return new WaitForSeconds(2);
44        generateQuestion();
45    }
46
47    void SetAnswers()
48    {
49        for(int i = 0; i < options.Length; i++)
```

```

...er\project_unity\PSM\Assets\Script\Quiz\QuizManager.cs
50     {
51         options[i].GetComponent<Image>().color = options
52         [i].GetComponent<AnswerScript>().startColor;
53         options[i].GetComponent<AnswerScript>().isCorrect = false;
54         //WrongPanel();
55         options[i].transform.GetChild(0).GetComponent<Text>().text = QnA
56         [currentQuestion].Answer[i];
57
58         if(QnA[currentQuestion].CorrectAnswer == i+1)
59         {
60             options[i].GetComponent<AnswerScript>().isCorrect = true;
61             options[i].GetComponent<Image>().color = options
62             [i].GetComponent<AnswerScript>().startColor;
63             //CorrectPanel();
64         }
65     }
66
67
68     void generateQuestion()
69     {
70         if (QnA.Count > 0)
71         {
72             currentQuestion = Random.Range(0, QnA.Count);
73
74             QuestionTxt.text = QnA[currentQuestion].Question;
75             FindObjectOfType<popOut>().PopExit();
76             SetAnswers();
77         }
78         else
79         {
80             Debug.Log("Out of Question"); //
81             FindObjectOfType<popOut>().PopExit();
82         }
83     }
84 }
85 }
86

```

Appendix 8: Script of Sign

```
D:\Users\Acer\project_unity\PSM\Assets\Script\Sign.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Sign : MonoBehaviour
6 {
7     public GameObject Tips;
8
9
10    private void OnTriggerEnter2D(Collider2D collision)
11    {
12        if (collision.tag == "Player")
13        {
14            Debug.Log("player in range");
15
16            Tips.SetActive(true);
17        }
18    }
19    private void OnTriggerExit2D(Collider2D collision)
20    {
21        if (collision.tag == "Player")
22        {
23            Debug.Log("player out of range");
24
25            Tips.SetActive(false);
26            Destroy(Tips);
27        }
28    }
29 }
30
```

Appendix 9: Script of Pop out

D:\Users\Acer\project_unity\PSM\Assets\Script\popOut.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class popOut : MonoBehaviour
7 {
8     public GameObject popUpBox;
9     public GameObject computer;
10    //public Text popUpText;
11    //public string popUp;
12    public static bool Paused = false;
13
14    // Start is called before the first frame update
15    private void Reset()
16    {
17        GetComponent<BoxCollider2D>().isTrigger = true;
18    }
19
20    // Update is called once per frame
21    public void OnTriggerEnter2D(Collider2D other)
22    {
23
24        if (other.tag == "Player")
25        {
26
27            Time.timeScale = 0f;
28            Paused = true;
29            popUpBox.SetActive(true);
30            Debug.Log("computer is triggered");
31
32        }
33        else
34        {
35            popUpBox.SetActive(false);
36
37        }
38    }
39
40
41    public void OnTriggerExit2D(Collider2D other)
42    {
43        Destroy(computer);
44    }
45
46    public void PopExit()
47    {
48        StartCoroutine(WaitForWhile());
49    }
```

D:\Users\Acer\project_unity\PSM\Assets\Script\popOut.cs

```
50
51    IEnumerator WaitForWhile()
52    {
53        yield return new WaitForSeconds(1);
54        popUpBox.SetActive(false);
55    }
56 }
57
```

2

Appendix 10: Script of waypoint

```
...object_unity\PSM\Assets\Script\wayPoint\WaypointNotFlip.cs 1
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class WaypointNotFlip : MonoBehaviour
6 {
7     [SerializeField] private GameObject[] waypoints;
8     private int currentWaypointIndex = 0;
9
10    [SerializeField] private float speed = 2f;
11
12    private void Update()
13    {
14        if (Vector2.Distance(waypoints[currentWaypointIndex].transform.position,
15                             transform.position) < .1f)
16        {
17            currentWaypointIndex++;
18            if (currentWaypointIndex >= waypoints.Length)
19            {
20                currentWaypointIndex = 0;
21            }
22            transform.position = Vector2.MoveTowards(transform.position, waypoints
23            [currentWaypointIndex].transform.position, Time.deltaTime * speed);
24        }
25    }
```

Appendix 11: Script of Enemy

```
D:\Users\Acer\project_unity\PSM\Assets\Script\Enemy\Enemy.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Enemy : MonoBehaviour
6 {
7     protected int decayAmount = 20;
8
9
10    private void Start()
11    {
12
13    }
14    private void Reset()
15    {
16        GetComponent<BoxCollider2D>().isTrigger = true;
17    }
18
19    protected void OnTriggerEnter2D(Collider2D collision)
20    {
21        if (collision.tag == "Player")
22        {
23            Debug.Log($"{name} Triggered");
24            FindObjectOfType<HealthBar>().LoseHealth(decayAmount);
25
26        }
27    }
28 }
29
30 }
31
```


Appendix 12: Script of Enemy AI

```
D:\Users\Acer\project_unity\PSM\Assets\Script\Enemy\EnemyAI.cs 1
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class EnemyAI : MonoBehaviour
6 {
7     public List<Transform> points;
8     public int nextID = 0;
9     int idChangeValue = 1;
10    //Speed of movement or flying
11    public float speed = 2;
12
13    private void Reset()
14    {
15        Init();
16    }
17    void Init()
18    {
19        //Make box collider trigger
20        GetComponent<BoxCollider2D>().isTrigger = true;
21
22        //Create Root object
23        GameObject root = new GameObject(name + "_Root");
24        //Reset Position of Root to enemy object
25        root.transform.position = transform.position;
26        //Set enemy object as child of root
27        transform.SetParent(root.transform);
28        //Create Waypoint onject
29        GameObject waypoints = new GameObject("Waypoints");
30        //Reset Waypoint position to root
31        //Make waypoint object child of root
32        waypoints.transform.SetParent(root.transform);
33        waypoints.transform.position = root.transform.position;
34        //Create two points (game object) and reset their position to waypoints ↗
35        //Make the points children of waypoint object
36        GameObject p1 = new GameObject("Point1"); p1.transform.SetParent ↗
37        (waypoints.transform); p1.transform.position = Vector3.zero;
38        GameObject p2 = new GameObject("Point2"); p2.transform.SetParent ↗
39        (waypoints.transform); p2.transform.position = Vector3.zero;
40
41        //Init points list then add the points to it
42        points = new List<Transform>();
43        points.Add(p1.transform);
44        points.Add(p2.transform);
45    }
46
47    private void Update()
48    {
```

```
47     MoveToNextPoint();
48 }
49 void MoveToNextPoint()
50 {
51     //Get the next point transform
52     Transform goalPoint = points[nextID];
53     //Flip the enemy transform to look into the point's direction
54     if (goalPoint.transform.position.x > transform.position.x)
55         transform.localScale = new Vector3(-1, 1, 1);
56     else
57         transform.localScale = new Vector3(1, 1, 1);
58     //Move the enemy towards to goal point
59     transform.position = Vector2.MoveTowards(transform.position,
60     goalPoint.position, speed * Time.deltaTime);
61     //Check the distance between enemy and goal point to trigger next point
62     if (Vector2.Distance(transform.position, goalPoint.position) < 0.5f)
63     {
64         //check if we are at the end of the line (make the change -1)
65         if (nextID == points.Count - 1)
66             idChangeValue = -1;
67         //check if we are at the start of the line (make the change +1)
68         if (nextID == 0)
69             idChangeValue = 1;
70         //Apply the change on the nextID
71         nextID += idChangeValue;
72         //nextID = nextID + idChangeValue
73     }
74 }
75 }
76
```

Appendix 13: Script of Trap Object

```
D:\Users\Acer\project_unity\PSM\Assets\Script\trapObject.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class trapObject : MonoBehaviour
6 {
7     int decayAmount = 20;
8     private void Reset()
9     {
10         GetComponent<BoxCollider2D>().isTrigger = true;
11     }
12
13     private void OnTriggerEnter2D(Collider2D collision)
14     {
15         if (collision.tag == "Player")
16         {
17             Debug.Log($"{name} Triggered");
18             FindObjectOfType<HealthBar>().LoseHealth(decayAmount);
19         }
20     }
21 }
22
```

Appendix 14: Script of Chase Player

```
...Acer\project_unity\PSM\Assets\Script\Enemy\chasePlayer.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class chasePlayer : MonoBehaviour
6 {
7     [Header("chasePlayer Attributed")]
8     [SerializeField] private float speed;
9     [SerializeField] private float range;
10    [SerializeField] private float checkDelay;
11    [SerializeField] private LayerMask playerLayer;
12    private Vector3[] directions = new Vector3[4];
13    private float checkTimer;
14    private Vector3 destination;
15    int decayAmount = 20;
16    private bool attacking;
17
18
19    private void OnEnable()
20    {
21        stop();
22    }
23
24    // Update is called once per frame
25    void Update()
26    {
27        //Move computer to destination only if attacking
28        if (attacking)
29            transform.Translate(destination * Time.deltaTime * speed);
30        else
31        {
32            checkTimer += Time.deltaTime;
33            if (checkTimer > checkDelay)
34                CheckForPlayer();
35        }
36    }
37
38    private void CheckForPlayer()
39    {
40        CalculaterDirections();
41
42        //Check if computer sees player
43        for (int i = 0; i < directions.Length; i++)
44        {
45            Debug.DrawRay(transform.position, directions[i], Color.red);
46            RaycastHit2D hit = Physics2D.Raycast(transform.position, directions
47                [i], range, playerLayer);
48
49            if(hit.collider != null && ! attacking)
```

...Acer\project_unity\PSM\Assets\Script\Enemy\chasePlayer.cs

```
49         {
50             attacking = true;
51             destination = directions[i];
52             checkTimer = 0;
53         }
54     }
55 }
56
57 private void CalculaterDirections()
58 {
59     directions[0] = transform.right * range; //right direction
60     directions[1] = -transform.right * range; //left direction
61     directions[2] = transform.up * range; //Up direction
62     directions[3] = -transform.up * range; //down direction
63 }
64
65 private void stop()
66 {
67
68     destination = transform.position; //set destination as current position
69     attacking = false;
70
71 }
72
73 private void OnTriggerEnter2D(Collider2D collision)
74 {
75     if(collision.tag == "Player")
76     {
77         FindObjectOfType<HealthBar>().LoseHealth(decayAmount);
78     }
79     // stop once when hits something
80     stop();
81 }
82 }
83 }
84
```

Appendix 15: Script of Reset Position

```
...object_unity\PSM\Assets\Script\Enemy\ResetPositionVirus.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ResetPositionVirus : MonoBehaviour
6 {
7     [SerializeField] private GameObject[] enemies;
8     private Vector3[] initialPosition;
9     // Start is called before the first frame update
10    private void Awake()
11    {
12        //save the initial position of the enemies
13        initialPosition = new Vector3[enemies.Length];
14        for(int i = 0; i < enemies.Length; i++)
15        {
16            if(enemies[i] != null)
17                initialPosition[i] = enemies[i].transform.position;
18        }
19    }
20    public void ActivateRoom(bool _status)
21    {
22        //Activate/deactivate enemies
23        for(int i = 0; i < enemies.Length; i++)
24        {
25            if(enemies[i] != null)
26            {
27                enemies[i].SetActive(_status);
28                enemies[i].transform.position = initialPosition[i];
29            }
30        }
31    }
32    private void OnTriggerEnter2D(Collider2D collision)
33    {
34        if (collision.tag == "Player")
35        {
36            ActivateRoom(true);
37        }
38        else
39        {
40            ActivateRoom(false);
41        }
42    }
43 }
44
```

Appendix 16: Script of Arrow Trap

```
D:\Users\Acer\project_unity\PSM\Assets\Script\Arrow\ArrowTrap.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class ArrowTrap : MonoBehaviour
6 {
7     [SerializeField] private float attackCooldown;
8     [SerializeField] private Transform firePoint;
9     [SerializeField] private GameObject[] arrows;
10    private float cooldownTimer;
11
12    private void Attack()
13    {
14        cooldownTimer = 0;
15
16        arrows[FindArrow()].transform.position = firePoint.position;
17        arrows[FindArrow()].GetComponent<EnemyProjectile>().ActiveProjectile();
18    }
19    private int FindArrow()
20    {
21        for (int i = 0; i < arrows.Length; i++)
22        {
23            if (!arrows[i].activeInHierarchy)
24                return i;
25        }
26        return 0;
27    }
28    private void Update()
29    {
30        cooldownTimer += Time.deltaTime;
31
32        if (cooldownTimer >= attackCooldown)
33            Attack();
34    }
35
36 }
37
38
39
```

Appendix 17: Script of Enemy Projectile

```
..\project_unity\PSM\Assets\Script\Arrow\EnemyProjectile.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class EnemyProjectile : Enemy //wil damage the player every time they
   touch
6 {
7
8     [SerializeField] private float speed;
9     [SerializeField] private float resetTime;
10    private float lifeTime;
11
12    //Damage
13    public void ActiveProjectile()
14    {
15        lifeTime = 0;
16        gameObject.SetActive(true);
17    }
18
19    private void Update()
20    {
21        float movementSpeed = speed * Time.deltaTime;
22        transform.Translate(movementSpeed, 0, 0);
23
24        lifeTime += Time.deltaTime;
25        if (lifeTime > resetTime)
26            gameObject.SetActive(false);
27    }
28
29    private void OnTriggerEnter2D(Collider2D collision)
30    {
31        base.OnTriggerEnter2D(collision); //execute logic from parent script
   first
32        gameObject.SetActive(false); // when this hits any objects deactivate
   arrow
33    }
34
35 }
36
```


Appendix 18: Script of Falling Platform

```
..\Assets\Script\FallingPlatform\FallingWithoutCondition.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class FallingWithoutCondition : MonoBehaviour
6 {
7
8     Rigidbody2D rb;
9     Vector2 initialPosition;
10    bool platformMovingBack;
11
12    // Use this for initialization
13    void Start()
14    {
15        rb = GetComponent<Rigidbody2D>();
16        initialPosition = transform.position;
17    }
18
19    void Update()
20    {
21        if (platformMovingBack)
22            transform.position = Vector2.MoveTowards(transform.position,
23                initialPosition, 20f * Time.deltaTime);
24
25        if (transform.position.y == initialPosition.y)
26            platformMovingBack = false;
27    }
28
29    void OnCollisionEnter2D(Collision2D col)
30    {
31        if (col.gameObject.name.Equals("Player") && !platformMovingBack)
32        {
33            Invoke("DropPlatform", 0.2f);
34        }
35    }
36
37    void DropPlatform()
38    {
39        rb.isKinematic = false;
40        Invoke("GetPlatformBack", 1.5f);
41    }
42
43    void GetPlatformBack()
44    {
45        rb.velocity = Vector2.zero;
46        rb.isKinematic = true;
47        platformMovingBack = true;
48    }
49 }
50
```

Appendix 19: Script of Climb

```
D:\Users\Acer\project_unity\PSM\Assets\Script\climb.cs
1 using UnityEngine;
2
3 public class climb : MonoBehaviour
4 {
5     private float vertical;
6     private float speed = 8f;
7     private bool isLadder;
8     private bool isClimbing = false;
9     private Animator anim;
10
11
12     [SerializeField] private Rigidbody2D rb;
13
14     void Update()
15     {
16         anim = GetComponent<Animator>();
17         vertical = Input.GetAxis("Vertical");
18
19         if (isLadder && Mathf.Abs(vertical) > 0f)
20         {
21             isClimbing = true;
22
23         }
24     }
25
26     private void FixedUpdate()
27     {
28         if (isClimbing)
29         {
30             rb.gravityScale = 2f;
31             rb.velocity = new Vector2(rb.velocity.x, vertical * speed);
32         }
33         else
34         {
35             rb.gravityScale = 4f;
36         }
37     }
38
39     private void OnTriggerEnter2D(Collider2D collision)
40     {
41         if (collision.CompareTag("Ladder"))
42         {
43             isLadder = true;
44             isClimbing = true;
45             //anim.SetBool("climb", true);
46         }
47     }
48
49     private void OnTriggerExit2D(Collider2D collision)
```

```
D:\Users\Acer\project_unity\PSM\Assets\Script\climb.cs
```

```
50     {
51         if (collision.CompareTag("Ladder"))
52         {
53             isLadder = false;
54             isClimbing = false;
55             //anim.SetBool("climb", false);
56         }
57     }
58 }
```

Appendix 20: Script of Sticky platform

```
D:\Users\Acer\project_unity\PSM\Assets\Script\StickyPlatform.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class StickyPlatform : MonoBehaviour
6 {
7     private void OnTriggerEnter2D(Collider2D collision)
8     {
9         if (collision.gameObject.name == "Player")
10        {
11            collision.gameObject.transform.SetParent(transform);
12        }
13    }
14    private void OnTriggerExit2D(Collider2D collision)
15    {
16        if (collision.gameObject.name == "Player")
17        {
18            collision.gameObject.transform.SetParent( null);
19        }
20    }
21 }
22
```

Appendix 21: Script of Player movement

```
... \project_unity\PSM\Assets\Script\Player\PlayerMovement.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class PlayerMovement : MonoBehaviour
6 {
7     private Rigidbody2D rb;
8     private BoxCollider2D coll;
9     private Animator anim;
10    private SpriteRenderer sprite;
11
12
13    private float dirX;
14
15    //ladder Variables
16
17    [SerializeField] private LayerMask jumpableGround;
18    [SerializeField] private float moveSpeed;
19    [SerializeField] private float jumpForce;
20
21    private enum MovementState { idle, running, jumping, falling }
22
23
24    [SerializeField] private AudioSource jumpSoundEffect;
25    MovementState state;
26
27
28    // Start is called before the first frame update
29    private void Start()
30    {
31        rb = GetComponent<Rigidbody2D>();
32        coll = GetComponent<BoxCollider2D>();
33        sprite = GetComponent<SpriteRenderer>();
34        anim = GetComponent<Animator>();
35    }
36
37    // Update is called once per frame
38    public void Update()
39    {
40        dirX = Input.GetAxis("Horizontal");
41
42        rb.velocity = new Vector2(dirX * moveSpeed, rb.velocity.y);
43        if (Input.GetButtonUp("Jump") && IsGrounded())
44        {
45            jumpSoundEffect.Play();
46            rb.velocity = new Vector2(rb.velocity.x, jumpForce);
47        }
48        UpdateAnimationState();

```

...\project_unity\PSM\Assets\Script\Player\PlayerMovement.cs

```
50     }
51
52
53     private void UpdateAnimationState()
54     {
55
56         if (dirX > 0f)
57         {
58             state = MovementState.running;
59             sprite.flipX = false;
60         }
61         else if (dirX < 0f)
62         {
63             state = MovementState.running;
64             sprite.flipX = true;
65         }
66         else
67         {
68             state = MovementState.idle;
69         }
70
71         if (rb.velocity.y > .1f)
72         {
73             state = MovementState.jumping;
74         }
75         else if (rb.velocity.y < -.1f)
76         {
77             state = MovementState.falling;
78         }
79
80         anim.SetInteger("state", (int)state);
81     }
82
83     private bool IsGrounded()
84     {
85         return Physics2D.BoxCast(coll.bounds.center, coll.bounds.size, 0f,
86             Vector2.down, .1f, jumpableGround);
87     }
88 }
89
```

Appendix 22: Script of Player life

```
...r\project_unity\PSM\Assets\Script\Player\PlayerLife.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class PlayerLife : MonoBehaviour
7 {
8     private Rigidbody2D rb;
9     private Animator anim;
10
11     [SerializeField] private AudioSource deathSoundEffect;
12     public GameObject Over;
13     public GameObject InsufficientCoin;
14     [SerializeField] private AudioSource GameOverSound;
15     [SerializeField] private AudioSource FailSound;
16     [SerializeField] private AudioSource BG;
17
18     // Start is called before the first frame update
19     private void Start()
20     {
21         rb = GetComponent<Rigidbody2D>();
22         anim = GetComponent<Animator>();
23     }
24
25     private void OnCollisionEnter2D(Collision2D collision)
26     {
27         if (collision.gameObject.CompareTag("Trap"))
28         {
29             Die();
30         }
31     }
32
33     private void OnTriggerEnter2D(Collider2D collision)
34     {
35         if (collision.gameObject.CompareTag("Trap"))
36         {
37             Die();
38         }
39     }
40
41     public void Die()
42     {
43
44         deathSoundEffect.Play();
45         rb.bodyType = RigidbodyType2D.Static;
46         anim.SetTrigger("death");
47         resetScore();
48
49     }
```

...r\project_unity\PSM\Assets\Script\Player\PlayerLife.cs

```
50 public void GameOver()
51 {
52     //1- Restart the scene
53     Over.SetActive(true);
54     Time.timeScale = 0f;
55     GameOverSound.Play();
56     BG.Pause();
57     //2- Reset the player's position
58     //Save the player's initial position when game starts
59     //when respawning simply reposition the player to that init position
60 }
61
62 public void insufficientCoin()
63 {
64     InsufficientCoin.SetActive(true);
65     Time.timeScale = 0f;
66     FailSound.Play();
67     BG.Pause();
68
69 }
70
71 public void restart()
72 {
73     SceneManager.LoadScene(SceneManager.GetActiveScene().name);
74     resetScore();
75     Time.timeScale = 1f;
76 }
77
78 public void resetScore()
79 {
80     itemCollect.COIN = 0;
81     PlayerPrefs.SetInt("score", itemCollect.COIN);
82 }
83 }
84
```

Appendix 23: Script of Health Bar

```
... \Acer\project_unity\PSM\Assets\Script\Player\HealthBar.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class HealthBar : MonoBehaviour
7 {
8     [SerializeField] private AudioSource HurtSoundEffect;
9     public Image fillBar;
10    public float health = 100;
11
12    //100health => 1 fill amount
13    //45 health => 0.45 fill amount
14
15    public void LoseHealth(int value)
16    {
17        //Do nothing if you are out of health
18        if (health <= 0)
19            return;
20        //play the sound effect
21        HurtSoundEffect.Play();
22        //reduce the health
23        health -= value;
24        //Refresh the UI fillBar
25        fillBar.fillAmount = health / 100;
26        //Check if your health is zero or less => Dead
27        if (health <= 0)
28        {
29            FindObjectOfType<PlayerLife>().Die();
30        }
31    }
32
33    }
34    //press enter to test the health decrease or not
35    //private void Update()
36    //{
37    //if (Input.GetKeyDown(KeyCode.Return))
38    //LoseHealth(25);
39    // }
40
41 }
42
```


Appendix 24: Script of item collect

```
...cer\project_unity\PSM\Assets\Script\Player\itemCollect.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class itemCollect : MonoBehaviour
7 {
8     public static int COIN = 0;
9
10    [SerializeField] public Text CoinText;
11    [SerializeField] private AudioSource collectionSoundEffect;
12
13    private void OnTriggerEnter2D(Collider2D collision)
14    {
15        if (collision.gameObject.CompareTag("coin"))
16        {
17            collectionSoundEffect.Play();
18            Destroy(collision.gameObject);
19            COIN += 10;
20            CoinText.text = " " + COIN ;
21        }
22    }
23
24    public void Deducecoin()
25    {
26        COIN -= 10;
27        CoinText.text = " " + COIN ;
28    }
29    public void Increasecoin()
30    {
31        COIN += 10;
32        CoinText.text = " " + COIN ;
33    }
34 }
35 }
36
```

Appendix 25 : Script of Checkpoint tutorial

```
...ity\PSM\Assets\Script\ (D)Checkpoint\CheckpointTutorial.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
6 public class CheckpointTutorial : MonoBehaviour
7 {
8     private AudioSource finishSound;
9     private Animator anim;
10    public int nextSceneLoad;
11
12    private bool levelCompeted = false;
13
14    // Start is called before the first frame update
15    void Start()
16    {
17        nextSceneLoad = SceneManager.GetActiveScene().buildIndex + 1;
18        finishSound = GetComponent<AudioSource>();
19        anim = GetComponent<Animator>();
20    }
21
22    private void OnTriggerEnter2D(Collider2D collision)
23    {
24        if (collision.gameObject.name == "Player" && !levelCompeted)
25        {
26            if (itemCollect.COIN >= 10)
27            {
28                anim.Play("win");
29                finishSound.Play();
30                levelCompeted = true;
31                FindObjectOfType<LoadingScreen2>().LoadScene();
32
33
34                if (SceneManager.GetActiveScene().buildIndex == 6)
35                {
36                    Debug.Log("You Win Game");
37                    FindObjectOfType<PlayerLife>().resetScore();
38                }
39                else
40                {
41                    Invoke("CompleteLevel", 2f);
42                    if (nextSceneLoad > PlayerPrefs.GetInt("levelAt"))
43                    {
44                        PlayerPrefs.SetInt("levelAt", nextSceneLoad);
45                        FindObjectOfType<PlayerLife>().resetScore();
46                    }
47                }
48
49                Debug.Log("Level " + PlayerPrefs.GetInt("levelAt") +
```

```

..ity\PSM\Assets\Script\Checkpoint\CheckpointTutorial.cs
    "Unlocked");
50     }
51     else
52     {
53         levelCompeted = false;
54         FindObjectOfType<PlayerLife>().insufficientCoin();
55     }
56 }
57
58 }
59
60 private void CompleteLevel()
61 {
62     SceneManager.LoadScene(nextSceneLoad);
63     FindObjectOfType<PlayerLife>().resetScore();
64 }
65
66 }
67
68

```

Appendix 26: Script of Not Enough Coin

D:\Users\Acer\project_unity\PSM\Assets\Script\NotEnoughCoin.cs

```

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.UI;
5
6 public class NotEnoughCoin : MonoBehaviour
7 {
8     [SerializeField] public Text CText;
9     // Start is called before the first frame update
10    void Start()
11    {
12        CText.text = " x " + itemCollect.COIN;
13    }
14
15 }
16

```

Appendix 27: Script of Camera Controller

```
...roject_unity\PSM\Assets\Script\Player\CameraController.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class CameraController : MonoBehaviour
6 {
7     [SerializeField] private Transform player;
8
9     // Update is called once per frame
10    private void Update()
11    {
12        transform.position = new Vector3(player.position.x, player.position.y,
13        transform.position.z);
14    }
15 }
```

Appendix 28: Script of NPC

```
D:\Users\Acer\project_unity\PSM\Assets\Script\Npc.cs
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class Npc : MonoBehaviour
6 {
7     [SerializeField] float speed;
8     [SerializeField] float targetPosition;
9     [SerializeField] Transform playerTransform;
10    Rigidbody2D myRigidbody2D;
11    Transform target;
12    // Start is called before the first frame update
13    void Start()
14    {
15        myRigidbody2D = GetComponent<Rigidbody2D>();
16        target = GameObject.FindGameObjectWithTag
17            ("Player").GetComponent<Transform>();
18    }
19    // Update is called once per frame
20    void Update()
21    {
22        TargetFollow();
23        flipSprite();
24    }
25
26    void TargetFollow()
27    {
28        if (Vector2.Distance(transform.position, target.position) >
29            targetPosition)
30        {
31            transform.position = Vector2.MoveTowards(transform.position,
32                target.position, speed * Time.deltaTime);
33        }
34    }
35
36    void flipSprite()
37    {
38        if(playerTransform.position.x > transform.position.x)
39        {
40            //Face right
41            //transform.localScale = new Vector3(1,1,1) <- cannot use because it
42            //will resize the gameObject become bigger
43            transform.localScale = new Vector3(Mathf.Abs
44                (transform.localScale.x),transform.localScale.y,transform.localScale.z);
45        }
46        else if(playerTransform.position.x < transform.position.x)
47        {
48            //face left
49            //transform.localScale = new Vector3(-1,1,1) <- cannot use because
50            //it will resize the gameObject become bigger
51            transform.localScale = new Vector3(-1* Mathf.Abs
52                (transform.localScale.x), transform.localScale.y,
53                transform.localScale.z);
54        }
55    }
56 }
```

Appendix 29: Script of Sign Not destroy

D:\Users\Acer\project_unity\PSM\Assets\Script\SignNotDestroy.cs

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5 public class SignNotDestroy : MonoBehaviour
6 {
7     public GameObject Tips;
8
9
10    private void OnTriggerEnter2D(Collider2D collision)
11    {
12        if (collision.tag == "Player")
13        {
14            Debug.Log("player in range");
15
16            Tips.SetActive(true);
17        }
18    }
19    private void OnTriggerExit2D(Collider2D collision)
20    {
21        if (collision.tag == "Player")
22        {
23            Debug.Log("player out of range");
24
25            Tips.SetActive(false);
26        }
27    }
28 }
29
```

Appendix 30: Script of Loading Scene

```
...project_unity\PSM\Assets\Script>Loading>LoadingScreen1.cs
1 using System.Collections;
2 using UnityEngine;
3 using UnityEngine.SceneManagement;
4 using UnityEngine.UI;
5
6 public class LoadingScreen1 : MonoBehaviour
7 {
8     public GameObject LoaderUI;
9     public Slider progressSlider;
10    public int index;
11    public void LoadScene()
12    {
13        StartCoroutine(LoadScene_Coroutine());
14    }
15
16    public IEnumerator LoadScene_Coroutine()
17    {
18        progressSlider.value = 0;
19        LoaderUI.SetActive(true);
20
21        AsyncOperation asyncOperation = SceneManager.LoadSceneAsync(1);
22        asyncOperation.allowSceneActivation = false;
23        float progress = 0;
24
25        while (!asyncOperation.isDone)
26        {
27            progress = Mathf.MoveTowards(progress, asyncOperation.progress,
28                Time.deltaTime);
29            progressSlider.value = progress;
30            if (progress >= 0.9f)
31            {
32                progressSlider.value = 1;
33                asyncOperation.allowSceneActivation = true;
34            }
35            yield return null;
36        }
37    }
```

Appendix 31: Table Function Acceptance Test

Test Scenario ID	Test Scenario	Test Case ID	Test Case	Test Data	Pre-condition	Post-condition	Test step	Expected Result	Actual Result	Pass /Fail	Note/ Action
TS01	Main menu Function	TC01 - 01	Check the Start button	Click on the Start button.	Players enter the game application.	Player enter tutorial level	i) Launch the game ii) Click on the Start button	Able to enter tutorial level.	Able to enter tutorial level.	Pass	
		TC01 - 02	Check the Credit button	Click on the Credit button		Player enter Credit page	i) Launch the game ii) Click on the Credit button	Able to enter the credit page.	Able to enter Credit page	Pass	
		TC01 - 03	Check the Level button	Click on the Level button		Player enter level page	i) Launch the game ii) Click on	Able to enter the Level Page	Able to enter Level page	Pass	

							the Level button				
		TC01 - 04	Check the Quit button	Click on the Quit button		The confirmation to quit the game popped out.	i) Launch the game ii) Click on the Quit button	Able to pop out the confirmation menu to quit the game application.	Able to pop out the confirmation menu.	Pass	
TS02	Credit page Function	TC02 - 01	Check the Back button	Click on the Back button	I) Players enter the game application. II) Player enter Credit page	Player exit the credit page and back to main menu	I) Launch the game II) Enter Credit page. III) Click on the Back button	Able to back to the main menu	Able to back to the main menu	Pass	
TS03	Level Page	TC03 - 01	Check the “<” button	Click on the “<”	I) Players enter the	Player back to	i) Launch the game	Able to back to main menu	Able to back to main	Pass	

	Function			button	game application. II) Player enter	main menu	ii) Enter level page iii) Click on the “<” button.		menu		
		TC03 - 02	Check the “Reset” button	Click on the “Reset” button	Level page	The level button that has unlocked after the player passes the level will be locked again.	i) Launch the game ii) Enter level page iii) Click on the “Reset” button.	Able to lock the level again except tutorial.	Unable to lock the level again except the tutorial.	Fail	i) Adding “PlayerPrefs.SetInt(“levelAt”, levelAt);” into the script to reset the level button. ii) Set the button to reset the level after clicking on it.
		TC03 - 03	Check the “Reset” button	Click on the “Reset” button		The level button that has unlocked	i) Launch the game ii) Enter level page	Able to lock the level again except tutorial.	Able to lock the level again except the tutorial.	Pass	

						after the player passes the level will be locked again.	iii) Click on the “Reset” button.				
		TC03 - 04	Check the Tutorial and other level button	Click on the tutorial and other level button		Players can go to the level that is selected.	i) Launch the game ii) Enter level page iii) Click on the tutorial or other level button.	Able to bring the player to the level selected.	Able to bring the player to the level selected.	Pass	
		TC03 -05	Check the level able to be locked before the player	Click on the level button except tutorial button		Player cannot click on the level that has locked.	i) Launch the game ii) Enter level page iii) Click on the tutorial	The button does not bring the player to the level selected.	The button brings the player to the level selected.	Fail	I) Change the LevelSelection script. II) Reapply the script and the button to the

			passes the level.				or other level button.				level management.
		TC03-06	Check the level able to be locked before the player passes the level.	Click on the level button except tutorial button		Player cannot click on the level that has locked.	i) Launch the game ii) Enter level page iii) Click on the tutorial or other level button.	The button does not bring the player to the level selected.	The button does not bring the player to the level selected.	Pass	
TS04	Confirmation function	TC04-01	Check the “Yes” button.	Click on the “Yes” button.	I) Players enter the game application. II) Players enter the main menu page and	Players exit the game application	i) Launch the game ii) Enter main menu page iii) Click on the “Quit” button. IV) Click on the “Yes”	Able to exit the game application	Unable to exit the game application	Fail	i) Change the script by adding Application.Quit(). ii) Set the “Yes” button and set MainMenu.exit

		TC04-02	Check the “Yes” button.	Click on the “Yes” button.	click on the quit button.		button.	Able to exit the game application	Able to exit the game application	Pass	
		TC04-03	Check the “No” button.	Click on the “No” button.	III) Player enter confirmation page		i) Launch the game ii) Enter main menu page iii) Click on the “Quit” button. IV) Click on the “No” button.	Able to back the main menu page.	Able to back the main menu page.	Pass	
TS05	Pause menu Function	TC05-01	Check did the Pause menu pop out when the player presses the	Press “ESC” button	I) Players enter the game application. II) Players	Pause menu pops out and the game application pauses.	i) Launch the game ii) Enter tutorial level iii) Press the “ESC”	Able to pop out the pause menu and pause the game application.	Able to pop out the pause menu but unable to pause the game	Fail	i) Change the script by adding Time.timeScale = 1f to the pause menu

			“ESC” button.		enter the tutorial level.		button.		application.		script.
		TC05 - 02	Check did the Pause menu pops out when the player presses the “ESC” button.	Press “ESC” button	III) Players press the “ESC” button.			Able to pop out the pause menu and pause the game application.	Able to pop out the pause menu and pause the game application.	Pass	
		TC05 - 03	Check the Resume button	Click on the Resume button.	I) Players enter the game applicatio n. II) Players enter the tutorial level. III)	Players can play the game application after pausing.	i) Launch the game ii) Enter tutorial level iii) Press the “ESC” button iv) Click on the Resume button. .	Able to play the game application after pause.	Able to play the game application after pause.	Pass	

					<p>Players press the “ESC” button.</p> <p>IV) Click on the Resume button.</p>						
		TC05 - 04	Check the Main Menu button	Click on the Main menu button.	<p>I) Players enter the game application.</p> <p>II) Players enter the tutorial level.</p> <p>III) Players press the</p>	Bring the players back to the main menu page.	<p>i) Launch the game</p> <p>ii) Enter tutorial level</p> <p>iii) Press the “ESC” button</p> <p>iv) Click on the Main menu button. .</p>	Able to go back to the main menu page.	Able to go back to the main menu page.	Pass	

					“ESC” button. IV) Click on the Main menu button.						
		TC05 - 05	Check the Quit button	Click on the Quit button.	I) Players enter the game applicatio n. II) Players enter the tutorial level. III) Players press the “ESC”	Exit the game application .	i) Launch the game ii) Enter tutorial level iii) Press the “ESC” button iv) Click on the Quit button. .	Able to exit the game application.	Able to exit the game application.	Pass	

					button. IV) Click on the Quit button.						
TS06	Player movement	TC06 - 01	Check the response on the jumping	Pressing the space button	Player enter and play the game	Players will enter a tutorial or other game level.	i) Launch the application ii) Play the game iii) Press the Space, left and right button to move the character	Player character is able to jump on the platform.	Player character jumps on the platform.	Fail	Add the Tilemap collider to the platform after the player character can jump on the platform.
		TC06 - 02	Check the response on the jumping	Pressing the space button				Player character is able to jump on the platform.	Player character is able to jump on the platform.	Pass	
		TC06	Check the	Pressing				Player	Player	Fail	Add the

		-03	response on running	the “W”, “S”, “A”, and “D” button				character able to move and climb on the platform	character moves left and right on the platform, up and down to climb the ladder.		Tilemap collider to the platform after the player character can move on it. .
		TC06-04	Check the response on running	Pressing the “W”, “S”, “A”, and “D” button				Player character able to move and climb on the platform	Player character able to move and climb on the platform	Pass	
		TC06-05	Check the animation on idle of player character	Start the game				The player character plays idle animation when standing.	Player character plays the idle animation when	Pass	

								standing.			
		TC06-06	Check the animation on jumping	Pressing the space button				The character will play the animation when jumping	Player do not play the falling animation after jumping	Fail	i) Find the problem ii) solve the problem after watching some video iii)exchange the animation with falling
		TC06-07	Check the animation on jumping	Pressing the space button				The character will play the animation when jumping	Player do not play the falling animation after jumping	Pass	
		TC06-08	Check the animation on	Pressing the space button				The character will play the animation	Play the jumping animation	Pass	

			jumping					when jumping	when jumping		
		TC06-09	Check the animation on running	Pressing the left or right button				The character will play the animation when running	Play the running animation when pressing the left and right button.	Pass	
TS07	Rewards & Punishment	TC07-01	Check the response on collecting coin(collect score)	Collecting coins	Players enter tutorials or other levels and collide with the coins,	Number of the coins collected increased.	i) Launch the application ii) Play the game iii) Press the Space, left and right button to move the character	Add 10 coins collected and show it on the coin counter.	The coin counter does not show the number of coins collected.	Fail	i) Add CoinText.text = " " + COIN ; into the script "itemCollect".
		TC07-02	Check the response on collecting	Collecting coins	computers and obstacles.				The coin counter does not show the correct	Fail	Add COIN += 10; in to the script "itemCollect".

			coin(collect score)				iv) move the character to collide with the obstacles coins and computers.		number of coins collected.		
		TC07-03	Check the response on collecting coin(collect score)	Collecting coins					The coin counter shows the correct number of coins collected.	Pass	
		TC07-04	Check the response colliding with the obstacles(reducing life)	Collide with the obstacles		Player life deduced and shown in the health bar.		Reduce 20 life in the health bar.	Reduce 20 life in the health bar.	Pass	
		TC07-05	Check the response	Collide with any		Restart the level		Restart level	Restart level	Pass	

			when colliding with the obstacles 5 times	obstacles 5 times							
		TC07-06	Check the response when colliding with computer	Collide with computer		The question panel pops out.		Pop out the question panel after colliding with the computer.	Pop out the question panel after colliding with the computer.	Pass	
		TC07-07	Check the response when answer question correctly	Answer the question correctly.		The button became green and increased the coin's number.	i) Launch the application ii) Play the game iii) Press the Space, left and right button to	The button became green and increased the coin's number.	The button did not become green color when answer correctly.	Fail	Add "GetComponent<Image>().color = Color.green;" into the AnswerScript.

							move the character iv) move the character to collide with the computers. V) answer the question correctly.				
		TC07-08	Check the response when answer question correctly	Answer the question correctly.		The button became green and increased the coin's number.	i) Launch the application ii) Play the game iii) Press the Space, left and right button to move the	The button became green and increased the coin's number.	The button became green and increased the coin's number.	Pass	

						<p>character</p> <p>iv) move the character to collide with the computers.</p> <p>V) answer the question correctly.</p>				
		TC07-09	Check the response when answer question wrongly	Answer the question wrongly.		<p>The button became red and deduced the coin's number.</p> <p>i) Launch the application</p> <p>ii) Play the game</p> <p>iii) Press the Space, left and right button to move the character</p>	<p>The button became red and decreased the coin's number.</p>	<p>The button did not become red when answered wrongly.</p>	Fail	<p>Add “GetComponent<Image>().color = Color.red;” into the AnswerScript.</p>

							iv) move the character to collide with the computers. V) answer the question wrongly.				
		TC07 -10	Check the response when answer question wrongly	Answer the question wrongly.		The button became red and deduced the coin's number.	i) Launch the application ii) Play the game iii) Press the Space, left and right button to move the character iv) move the	The button became red and decreased the coin's number.	The button became red and decreased the coin's number.	Pass	

							character to collide with the computers. V) answer the question wrongly.				
TS08	Moving Platform , Obstacle movement	TC08-01	Check the animation and the movement of the moving platform	Start the game	Player enter the game and play the game	The animation and movement of the moving platform are moved to the correct position.	i) Launch the game ii) Play the game	Play the animation and move to the place that is set	Move to the place that is set but did not play the animation	Fail	i) Find the problem. ii) Solve the problem by editing the animation of the moving platform
		TC08-02	Check the animation and the movement of the	Start the game				Play the animation and move to the place that is set	Play the animation and move to the place that is set	Pass	

			moving platform								
		TC08-03	Check the animation and the movement of the obstacles	Start the game				Play the animation and move to the place that is set	Move to the place that is set and play the animation correctly	Pass	
TS09	Movement sound and background music	TC09-01	Check the jump sound response after jumping	Press the space button	Players enter the main menu and get into tutorial or other game levels.	Play jump sound when the player presses the space button.	i) Launch the application ii) Play the game iii) Press the Space button to move the character iv) move the	Play the jump sound after pressing the space button.	Played the jump sound after pressing the space button.	Pass	
		TC09-02	Check the hurt sound response	Collide with the obstacles		Play hurts sound when the		Play the hurt sound after colliding with	Played the hurt sound after	Pass	

			after losing life			player collides with the obstacles.	character to collide to the obstacles and food	the obstacles colliding with the obstacles			
		TC09-03	Check the dead sound response after hunger bar become 0	Collide with the obstacles until the hunger bar becomes 0		Play dead sound when the health bar of the player character becomes 0.		Play the dead sound after colliding with the obstacles and the hunger bar becomes 0.	Played the dead sound but the dead animation was extended and the level did not restart.	Fail	i) Find the problem ii) Solve the problem by adding “deathSoundEffect.Play();” to the PlayerLife script.
		TC09-04	Check the dead sound response after hunger bar become 0	Collide with the obstacles until the hunger bar becomes		Play dead sound when the health bar of the player character		Play the dead sound after colliding with the obstacles and the hunger bar becomes 0.	Play the dead sound after colliding with the obstacles and the	Pass	

				0		becomes 0.			hunger bar becomes 0.		
		TC04 -06	Check the backgroun d music of each level								

Appendix 31: User Acceptance Test

Smell Phishy (2D Platform Game) UAT

Dear all,
This is a **User Acceptance Test** for the **SMELL PHISHY (2D Platform Game)**. Smell Phishy (2D Platform Game) is an education game that focuses on Phishing awareness.

This UAT is used to fulfill the requirement for my Final Year Project (FYP) for the Bachelor of Computer Science (Graphic and Multimedia Technology). Do take part in this testing stage.



There are 2 sections of the UAT and the first one will need to answer the question that about functionality of the game. The second section will be going to ask about usability of the game.

1. Go to this link to download the game: [Smell Phishy by Teng123 \(itch.io\)](#).
2. Download Now > It's Free!!!!
3. Extract the file> Double clicks on the Smell Phishy.exe
4. Play the game.
5. Complete the User Acceptance Test (UAT)

Thank You!!!

CD19101 NG MAN TENG

Supervised by:
TS. DR. NOR SARADATUL AKMAR BINTI ZULKIFLI
FACULTY OF COMPUTING
UNIVERSITI MALAYSIA PAHANG

 mteng7263@gmail.com (not shared) [Switch accounts](#) 

*Required

DEMOGRAPHIC

Institution *

IPTA/IPTS

POLITEKNIK

DEMOGRAPHIC

Institution *

IPTA/IPTS

POLITEKNIK

SECONDARY SCHOOL

Other: _____

Institution Name *

(Example: Faculty of Computing, UMP)

Your answer _____

Role *

Lecturer / Teacher

Student

Other: _____

Gender *

Female

Male

Other: _____

Age *

(Example: 20)

Age *

(Example: 20)

Your answer

Have you studied Phishing awareness? *

Yes

No

User Acceptance Test (UAT)

Section 1: Functionality of the game

Did the "Main Menu" functional well? *

Strongly Disagree 1 2 3 4 5 Strongly Agree

Did the "Level Page" functional well? *

Strongly Disagree 1 2 3 4 5 Strongly Agree

Did the "Credits Page" functional well? *

Strongly Disagree 1 2 3 4 5 Strongly Agree

Did the "Credits Page" functional well? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the "Question Page" functional well? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the player character in the game function well? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the player character in the game function well? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the NPC in game follow the player character and chat with the player? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the Question panel pop out and function well when the player character collides with the computer in the game? *

Did the Question panel pop out and function well when the player character collides with the computer in the game? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the health bar of the player character function well? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the number of coins in the game increase and decrease correctly? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the enemies and traps function well in the game? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the light and neon system function well in the game? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the player character stick on the moving platform after the player jumped on? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Was the falling platform able to fall after the player character jumped on? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did the checkpoint jump to the next scene when the player fulfilled the requirement? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Able to climb the ladder? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

What is the error and problem that is still in the game?

Your answer

Section 2: Usability of the game

Do you like the Smell Phishy game? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Do you enjoy playing the Smell Phishy game? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Did you gain information when playing the game? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Do you want to play the Smell again game again? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

Do you think the Smell Phishy can help educate on Phishing attack awareness? *

	1	2	3	4	5	
Strongly Disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Strongly Agree

What is your comment(s) on the game?

Your answer

Appendix 33: Page in Itchi.io and QR code to download

