INTEGRATING GAME REPLAY VALUE TO ROLE-PLAYING FANTASY GAME : DUNGEON REVIVE

MUHAMMAD AMZAR RAIF BIN AMIR RASID

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

INTEGRATING GAME REPLAY VALUE TO ROLE-PLAYING FANTASY GAME : DUNGEON REVIVE

MUHAMMAD AMZAR RAIF BIN AMIR RASID

This report is submitted in partial fulfillment of the requirements for the Bachelor of Information Technology (Game Technology) with Honours

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2020/2021

DECLARATION

I hereby declare that this project report entitled

INTEGRATING GAME REPLAY VALUE TO ROLE-PLAYING FANTASY GAME : DUNGEON REVIVE

is written by me and is my own effort and that no part has been plagiarized without citations.

Student :  Date : 9.9.2021

(MUHAMMAD AMZAR RAIF BIN AMIR RASID)

I hereby declare that I have read this project report and found this report is sufficient in term of the scope and quality for the award of Bachelor of Information Technology (Game Technology) With Honours.

Supervisor : 

Date : 11 September 2021

(PROF. DR. SAZILAH BINTI SALAM)

DEDICATION

To all those who have supported, encouraged, challenged and inspire me and specially to my beloved parents, honourable lecturers and friends for all their guidance, love and attention which has make it possible for me to make it up this point

.

ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Sazilah Binti Salam for giving assistant to complete this project successfully. I heatedly thankful my lecturers and fellows who help me a lot during my project development process.

I am also thankful from the core of my heart to my beloved parents who always support and love me.

Thanks to all concerns.

ABSTRACT

This project focus on developing a game that integrated with game replay values. This game was developed to keep the player entertained and keep coming back to play the game after the first completion. The genre is rogue-like which is a sub-genre to role playing game(RPG) genre, where the levels are randomly generated and progress of player character after death is lost. Due to the fact that many other games in the same genre having the same issues with the game replay value, most of the game is fun to play but not many that people always play the game again for more experiences. This project is developed to design systems or mechanics to keep player playing the game without feel bored. Thus, the outcome for this project is to design systems and mechanics to integrate it with the game to enhance the game replay values.

ABSTRAK

Projek ini memberi tumpuan untuk mengembangkan permainan video yang disatukan dengan nilai permainan semula. Permainan video ini dikembangkan untuk menghiburkan pemain dan terus kembali bermain untuk setelah selesai pertama kali selesai bermainan. Genre rogue-like merupakan sub-genre kepada genre role playing game (RPG) , di mana tahap permainannya dihasilkan secara rawak dan kemajuan watak pemain setelah kematian akan hilang. Oleh kerana banyak permainan video lain dalam genre yang sama mempunyai masalah yang sama dengan nilai permainan semula, kebanyakan permainan video ini menyeronokkan untuk dimainkan tetapi tidak banyak yang membuat orang rasa untuk bermain semula untuk lebih pengalaman. Projek ini dibangunkan untuk merancang sistem atau mekanik agar pemain terus bermain tanpa merasa bosan. Oleh itu, hasil untuk projek ini adalah merancang sistem dan mekanik untuk mengintegrasikannya dengan permainan video untuk meningkatkan tahap nilai permainan semula.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. Project Background

This project is to develop a game that integrated with game replay values. The game main focus is to develop systems that enhance game replay values. The game developed is focused on Rogue-lite genre. Rouge-lite is a sub-genre of Role Playing game (RPG) genre but the levels is procedurally generated, having permanent death of player's character and progression mechanics. Gameplay, player controls character to escape from the dungeon. Played on orthographic view. Player can upgrades abilities that suit their play to survive through the dungeon.

1.2. Problem Statements

The problem with other video games in the same genre is most of them are lacking of the game replay values depends on how the game is designed. After the first completion the game, it lost all of it value and not interesting to play anymore because the players already knew what will happen. So how the game can be designed to have the game replay value attraction is very important.

Replay value is a term used to assess a video game's potential for continued play value after its first completion. Factors that influence replay value are the game's extra characters, secrets, ability system, alternate endings or achievements. The replay value of a game may also be based entirely on the individual's tastes.

1.3. Objective

This project embarks on the following objectives:

i. To investigate how existing games integrate game replay values.
ii. To develop a Role-Playing fantasy game that has game replay value attraction.
iii. To evaluate the video game potential for continued to be played after the first completion based on player experience.

1.4. Goals and Genre

The game about an adventurer named Atlas who died in a dungeon but somehow, he was brought back to life. Soon he knows that he is trapped inside the dungeon and start to muster his skills to escape from the dungeon. The main goal of the game is to escape from the dungeon. This project is a rouge-like video game set in a fantasy world.

Rouge-lite is a sub-genre of Role-Playing games (RPG) genre where player must go through all the procedural generated levels and player's character having permanent death each run. Each time players play the game; they will face new challenges as the layout of the levels are randomly generated.

1.5. Game Features

The target groups for the game is for player who like challenges and able to improve their skill as longer they play the game.

Player have to survive and find the way out of the dungeon. Find the boss of each level to advance to the next level. Player can upgrade their skills to increase the chance of surviving longer through the dungeon.

1.6. Conclusion

The expected outcome from the development of the game is to produce a game that able to attract people to keep playing the game and being addictive. Thus, the study can investigate the problems on the game replay value system for future recommendation on video game development.

CHAPTER 2

LITERATURE REVIEW AND PROJECT METHODOLOGY

2.1. Introduction

This chapter will discuss on the genre of the game, the list of existing games that are related to the project, comparison of the existing games and the project methodology.

2.2. Genre

Rouge-lite is a sub-genre of Role-Playing games (RPG) genre where player must go through all the procedural generated levels and player's character having permanent death each run. Each time players play the game; they will face new challenges as the layout of the levels are randomly generated. Most rogue-lites are based in fantasy world, mostly influenced from tabletop role playing games such as Dungeons & Dragons (DND) .

Meanwhile rogue-lite is slightly different, it focus on reaching the end and not on the run itself and also has carryover between the runs to further improve the player's character.

Every game has and an endpoint, where player defeat the final boss or survive through difficulties, but for rouge-likes is not focus on reaching the end but seeing what happens on the next run, as the next run might be better or worst depend on player's luck.

2.3. Existing Games

1) Hades

Hades is a rogue-lite action dungeon crawler video game developed and published by Supergiant Games. In this game, player play as the Underworld Prince, Zagreus in attempt to escape from underworld which is governed by his father, Hades. Player must survive through all three regions of underworld with the blessings of other gods of Greek mythology before defeating Hades. After defeating Hades, Zagreus find out that he can not survive long on the surface of the earth, so keep on escaping the underworld.

Figure 2.1 Hades logo

- Play as Zagreus to escape from the underworld by surviving through all the regions and bosses.

Figure 2.2 Mirror upgrades

- Use a unique currency to permanently upgrade to help player get stronger and add more chances that benefit the player.

Figure 2.3 Choosing equipment

- Player can choose any weapon and keepsake that they want before starting the run.

Figure 2.4 Weapon upgrades

- As player play the game, the weapon will unlock new abilities and each abilities can be upgraded.

Figure 2.5 Achievement list

- The Fated List, in other word is a list of achievement of what player can achieve and get reward from.

Figure 2.6 Achievement list

- Pact of punishment, where player can increase the difficulty of the game to get more rare rewards.

2) Enter The Gungeon

Enter the Gungeon are developed by Dodge Roll and published by Devolver Digital in 2016. Enter the Gungeon is a dungeon crawler with a challenging battle and evolving series of floors filled with dangerous bosses. The game follow a band of misfits that descend into the dungeon to find a time machine that have power to travel through time. This game focus on player skill to shoot and dodge bullet.

Figure 2.7 Enter the Gungeon logo

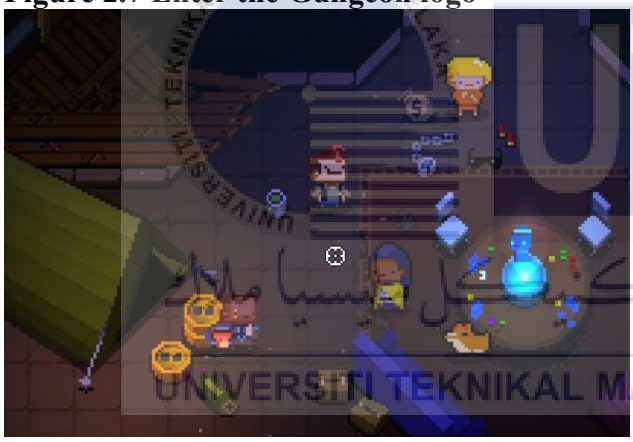


Figure 2.8 Character selection

Figure 2.9 Different ability

- Player can select one from four characters which is named Marine, Convict, Pilot and Hunter. All the characters each having different special abilities that player can use during the run.

Figure 2.10 Health point and economy

- Health point is can not be healed while in the run, can only be upgraded by progressing through the dungeon and buying heal point from a merchant. To buy item from the merchant, player can only use the currency that player accumulates by killing enemies. The upgrades and the economy is not permanent and lost after player exit the dungeon.

Figure 2.11 Level generation

- The levels are procedurally generated, player need to clear each room to proceed to the next room. Some cleared room enable teleportation to other room. Player can choose to fight the boss early if the boss room founded earlier or just grind for more currency to buy from the shop.

2.4. Comparison of Existing Games

Table 2.1 Comparison of existing games

	Hades	Enter the Gungeon
Gameplay	Play as Zagreus as player battles and survives through three underworld regions and to vanquish Hades with the help of Olympian gods' blessings. Every time player clears an encounter gain a new strength or reward.	Battle and survive through the dungeon. The dungeon is procedurally generated. Discover and gather precious loot that can help player to gain advantages over enemies.
Game Mechanics	Choose different weapons and keepsakes, each weapon and keepsakes can be upgraded to be more powerful. Use combination of weapon and blessing from the gods to battle and survive through the levels. Have currency that lost after the player death and unique currency that carried over to be used for upgrades.	Choose different characters, each character have different abilities such as calling support and lockpicking chests. Shoot and dodge bullet efficiently. Find precious weapon and loots or buy from the merchant. All the economy gain and loots is lost after player death but not for some unique items like a potion to increase player's total health point.
Game Replay Values	Player growth, player start with the default ability and as player complete each run, player can use unique currency which is carried over by the player to upgrades weapons or abilities. Further increasing the chance of player surviving throughout the underworld regions. Upgrades that have increase in statistic is permanent while blessing from the gods is lost after death. Story progression, happen in certain numbers of ways,the most easy is each time player died it unlocks new story progression or depending on how far player can survive it can unlock new story progression or unlock new weapon aspect which is a new way player can use the weapons.	Each time player play the game the dungeon rooms, the spawned enemies, the loots and bosses is completely procedurally generated. So each times is a new experience and not the same from the previous run. Enter the Gungeon is a game focused on player skill instead of growth of the characters. Player can even win the game by facing with enemies or bosses using the basic gun. But each enemies and bosses have different attack pattern so player have to train their skill to adapt to each enemies.

	Pact of punishment, after player first completion of the game, it unlock a new system called “Heat system” where player can manually increase the difficulty of the game. At each level of difficulty player can get rare rewards that only dropped after defeating the bosses. So player can get new experience by player the game while increasing the difficulty. Achievement list where player can see any challenges that player can set what to aim during a run to get more rewards. Cosmetics or decoration, with abundant amount of resources player can use buy house decoration or even modify how the game user interface looks.	Player can experience over 300 different guns and item that can be combined to achieve more powerful effects called "Synergies”. As player progress through multiple runs, player unlock new characters and rewards from the bosses can be used for some permanent upgrades.

- For the proposed project Dungeon Revive, player play as a wizard who is trapped inside the dungeon. Player must survive through the entire dungeon, defeat the bosses and escape from the dungeon.

- The first time playing the player is very weak. As player keep playing the game, player can gain more coins that drop from the enemies that to be used to upgrade health, speed and magic travel time so that player can get stronger if player has enough coins.

- Player also have achievement system where the list of available challenges that player have to complete to gain more rewards for upgrades. Also after the first completion of the game, the game will gain more difficulty and player have to spend more time defeating enemies and bosses.

2.5. Project Methodology

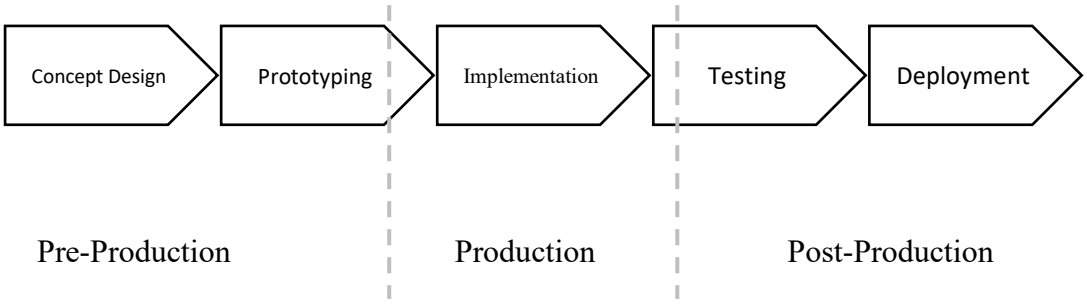


Figure 2.12 Game Development Life Cycle (GDLC)

- **Concept Design**
  Brainstorming process on the game replay value and creating the design for mechanic. Planning concept design for the level environments, enemies, character and user interface.

- **Prototyping**
  Creating the software prototype for the game, blocking the world environment design and preview on the game replay value mechanics. Creation of the game assets.

- **Implementation**
  The process on programming the source codes for game mechanics, game replay value systems, player actions, enemy Artificial Intelligent (AI), currency system, progression mechanics and winning conditions.

- **Testing**
  Playable alpha and beta version of the game is tested for improvement. The game is tested by small group of testers. Feedback and information is collected from the response of the testers.

- **Deployment**
  Where full complete version of the game which also known as master version is ready to be released to the market to sell to the public or endorse publicly as downloadable free game.

2.6. Conclusion

In the nutshell, this chapter explain why I choose certain type of genre for my game and why it is suitable for the development. I also provide the comparison of my game between the same type of game genre that available in the market. I also provide the methodology use in development of the project which is GDLC .

CHAPTER 3

ANALYSIS

3.1. Requirement Analysis

3. 1. 1. Project Requirement

TABLE 3.1 Existing Game Analysis

Game	Hades	Enter the Gungeon
Player roles	Battle and survive each encounters with enemies to reach the end. Player can choose to master 6 type of weapons, each aspects of the weapons and upgrade it. Player have to use temporary power-ups that gained from clearing each encounter to help gain advantages fighting against enemies.	Shoot and dodge bullet. Player required to improve their skill to dodge enemy bullets as the game required player to constantly dodge bullet. Player has to practice their timing evading things. Player choose to play up to 4 characters each having different special abilities.
Gameplay	Player required to focus on player skill on using 6 different kind of weapon (sword, spear, shield, bow, gauntlet and gun) and combinations of power ups. Player can also focus on collecting resources for upgrades that increase the chances of player surviving through or increase the chances to get better rewards or power-ups. Use combination of power-ups and weapon to battle and upgrade it through the run. Choosing the option that benefits and suit the kind of player play	Player require to dodge bullets. Player have to find defeat the boss to progress. Defeating a boss unlock new things that player can use for permanent upgrades. Player also focused on finding good weapons and collection resources, then the resources can be used to buy heal or power-ups. Player involve in exploration of the dungeon and using surrounding to block and evade enemy attacks.

	style. Player progressing through each regions, the enemies also will get harder. Each boss and enemies having different attack pattern and may vary on level of the "Heat" from the pact of punishment.	
Victory condition	Clearing the enemies in a encounter then player can proceed through. Player defeating the three bosses of different regions and defeating Hades at the surface of the underworld.	Player defeating the last Boss. The last boss can be different each time player advance to the final boss fight location.
Core Mechanic	Slashing, dashing, thrusting, backstabbing, aiming, maneuver, buying boost, trading, selling power-ups, convert resource, difficulty modifier, power upgrades, weapon and item upgrades, NPC interaction.	Shooting, dodging, aiming, maneuvering, upgrading, shop purchasing.
Progression	Player progress through regions of the underworlds like Tartarus, Asphodel, Elysium and Temple of Styx. Kill bosses of every region give player resources for permanent upgrades and some resources accumulated through the run also be used for upgrades. Every time player dies the game unlock a new story progression and player advance to certain encounter with certain requirement may unlock new story progression or weapon aspect.	Player killing boss unlock new room and get reward for permanent upgrades.

User Interface Feature	Health UI, resources UI, weapon, item and status upgrade UI, shop UI	Health UI, resources UI, equipment UI, teleportation UI
Camera Models	Orthographic camera	Orthographic camera
Storyline	Player play as Zagreus in his attempt to escape from the depth of the underworld with help from Olympian gods. Player unlock new story progression every time player dies. Player can interact with every NPC in the game. The main story progress through player completing the game ten times. Side story	Aan enormous bullet fell from space and crashed into Gunymede's surface, and destroyed a fortress in the process. Its resulting magic subsequently created a time machine of immeasurable power; a gun that can kill the past.The fortress was rebuilt with the highest of security measures to guard the gun, and adventurers known as "gungeoneers" hailed from places over the galaxy to claim their chance at changing their past. The player can play as eight gungeoneers, each with their own stories and regrets, as they decide to enter the fortress and descend into the Gungeon to find the gun in order to correct their wrongs.

3. 1. 2. Technical Requirement

- **Mouse and Keyboard**

  Mouse and keyboard are the traditional input devices for pc gaming that are available since the earlier time of the industry. Mouse and keyboard provide more dexterity and precision in the gameplay. Keyboard allows more hotkeys management to be implement that ease the player by modify suitable input that fit their preferences

3. 1. 3. Software Requirement

- **Unity Engine**

  For the game development, creating prototype and final product.

- **Adobe Photoshop**

  Designing and creating concept art and level design also creating of game assets.

- **Visual Studio**

  Tool to programming all the mechanics.

3. 1. 4. Hardware Requirement

- **Laptop**
- **Mouse**
- **Monitor**
- **Wacom drawing tablet**

3.2. Project Schedule And Milestone

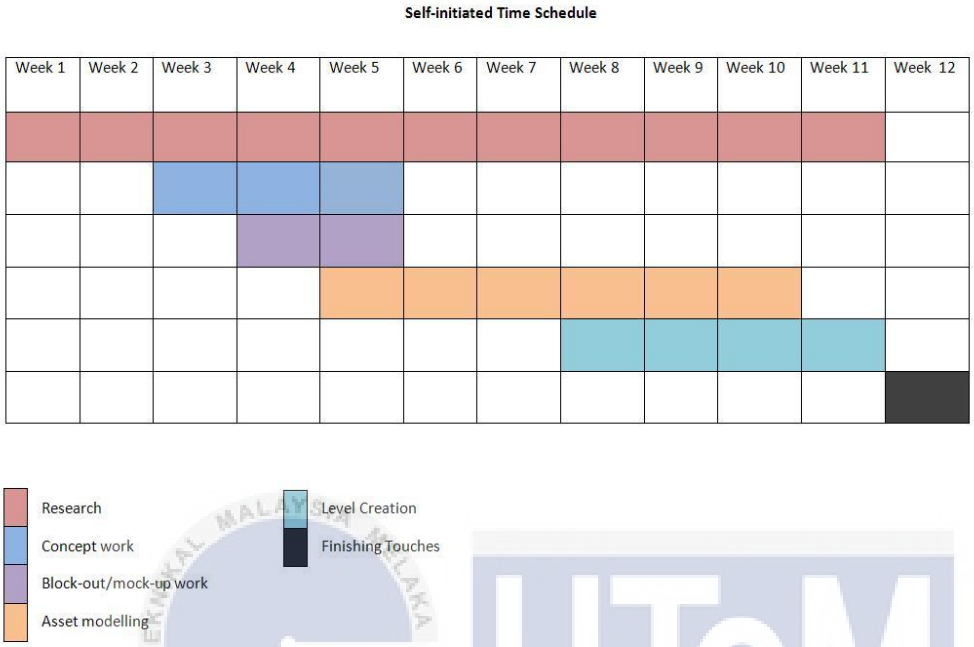


Table 3.2 Project Gantt Chart

- **Research**

Research on similar games and article on research topics.

- **Concept Work**

Creating concept art, level design and characters design

- **Block-out**

Process of blocking out the environment

- **Asset Creation**

Creating assets and sprites for the games

- **Level Creation**

Program the level progression, level transition and mechanics.

- **Finishing Touches**

Code all the intended mechanics.

3.3. Conclusion

- This chapter was focus on analyzing the differences of the similar game that are related to the propose project and identifying technical requirement, software requirement, hardware requirement and other requirement. Project schedules and milestones also explained in this chapter.

CHAPTER 4

DESIGN ANALYSIS

4. 1. Introduction

Game design is one of the most important process in video game development and the crucial phase in Dungeon Revive development phase. It involved game architecture and game design that can be divided into many parts such as gameplay, core mechanics, flowboard, level progression, storyline, user interface or interaction model, game art, game world, character design, camera design, audio and sound effect.

4. 2. Game Architecture

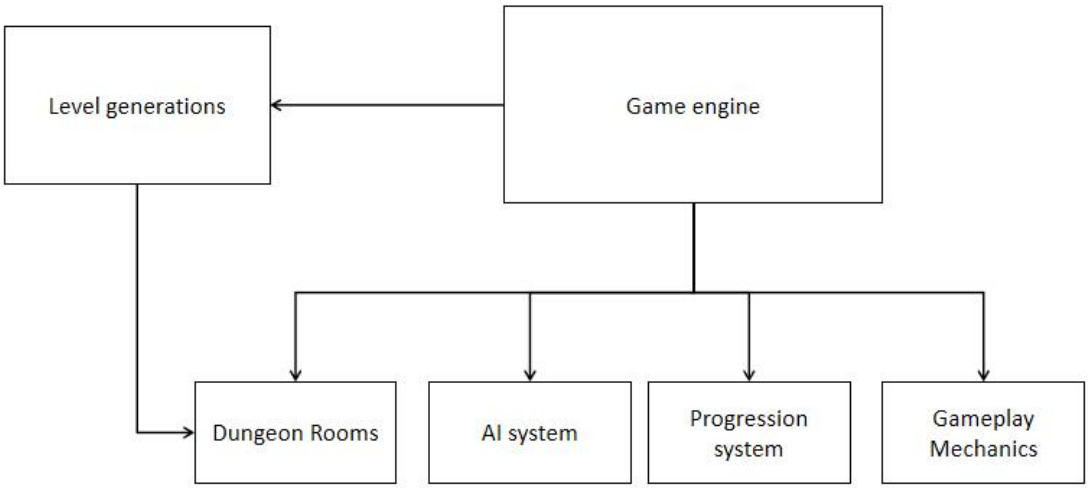


Figure 4.1 Rogue-like Game Architecture

4. 3. Game Design

4. 3. 1. Gameplay

- **Player Roles**

  Improving character growth and player skills, player focus on playing the game efficiently and improving player skill to maneuvering player's character to dodging enemy attacks and upgrading the character growth.

- **Rules**

i. Player have to advance through each room to proceed to next level.
ii. Player need to have enough resources for upgrading the character's status.
iii. Player need to complete challenges or certain requirement to unlock rewards or new mechanics.
iv. Player lost progression if player die, but the resources is remained.

- **Victory Conditions**

  Player have to defeat the boss of each region to progress to next region. Defeating the last boss to achieve main goal.

- **Termination Conditions**

  When player health point hit zero, player dies and have to restart again from the start at the beginning of the dungeon.

- **Level Of Difficulty**

  The difficulty of the game get improved everytime player complete the game. The game difficulty is upgraded, the enemies is harder to kill and the bosses have new attack patterns.

4. 3. 2. Core mechanics

- **Health system**

  Player have limited amount of health that can be upgraded to increase the amount.

- **Shooting mechanics**

  Player can shoot magic bullet that disappear after short amount of time. Player can shoot enemies bullet to deflect it.

- **Economy system**

  Every time player kills an enemy gain a random amount of coins that can be used in the shop to buy permanent upgrades.

- **Maneuvering**

  Player can move freely and teleport ahead to dodge bullets. With perfect timing player can teleport through walls.

- **Defence mechanics**

  Player can cast temporary magic shield or use the surrounding environment to block enemies attacks.

- **Enemy AI**

  There is two types of enemies, ranged and melee .Each region have different kind of enemies according to the environment. The more progress player made the stronger the enemies.

- **Progression mechanics**

  The more player play the game the more resources that player have and can upgrade to get stronger. As player complete the game player unlock new mechanics where player can manually change the difficulty.

- **Collectibles**

  The enemies that player kill can randomly drop heal point that player can collect.

- **Score**

  The numbers of enemies killed is counted to complete certain challenges.

4. 3. 3. Flowboard

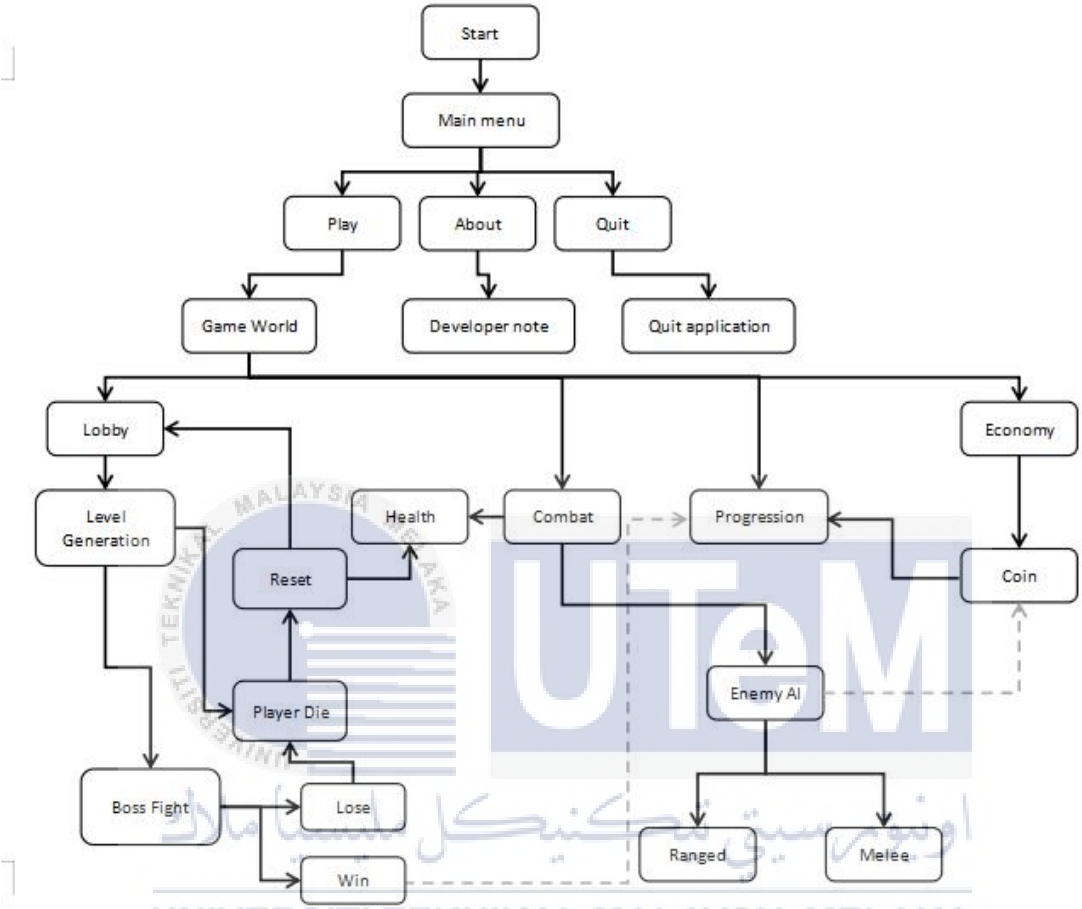


Figure 4.2 Game flowboard

4. 3. 4. Level progression

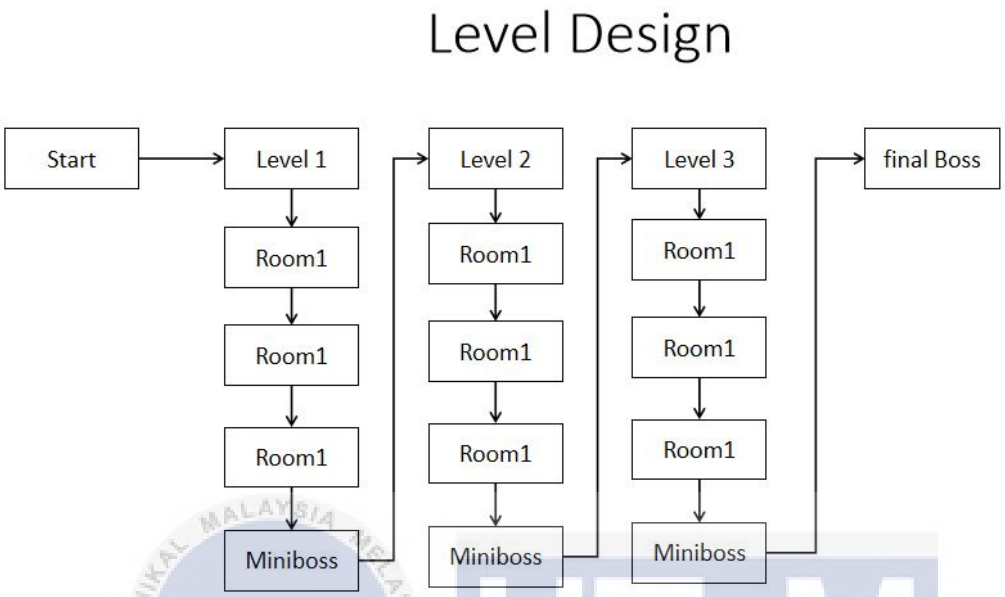


Figure 4.3 Level Design

Early Stage(When starting new game)

- **Condition**

Low health, slow speed, bullet can only travel for a short time and initial amount of coins is low.

- **Result**

Have a hard time fighting enemy.

Late Stage(After couple of times playing)

- **Condition**

Lot of health, faster movement speed, bullet can travel for a long time and
Amount of coins is a lot and can be use to buy upgrades.
Better understanding of the enemy weakness and game mechanics

- **Result**

Can easily kill enemies.

4. 3. 5. Storyline

The game about a wizard named Atlas who died in a dungeon but somehow, he was brought back to life. Soon he knows that he is trapped inside the dungeon and start to muster his skills to escape from the dungeon. The main goal of the game is to escape from the dungeon. After defeating the final boss of the game, he escaped…but later he find out that the dungeon is revived and the difficulty is more harder than what he first experienced.

4. 3. 6. User interface / Interaction Model

- **Earlier sketch of UI and GUI before implementation**

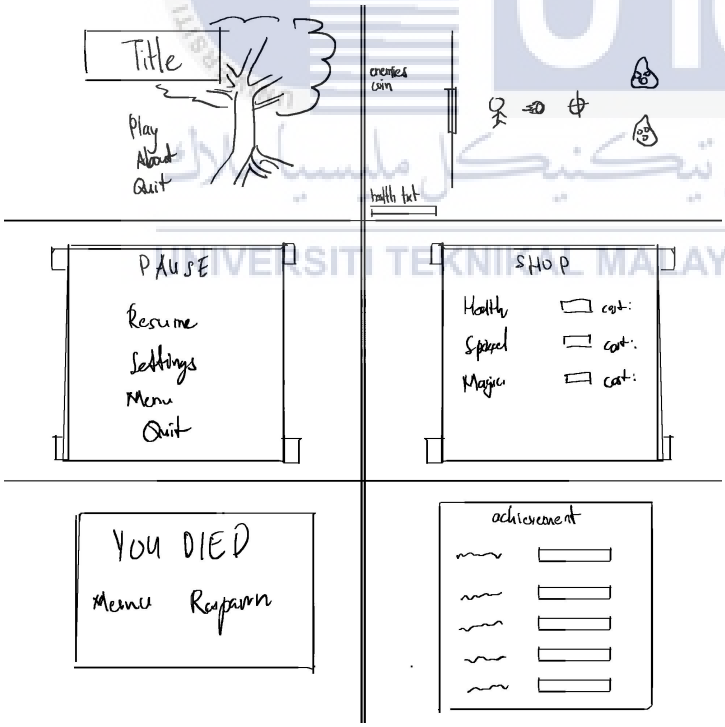


Figure 4.4 UI Sketch
Sketching for the user interface is made in Adobe Photoshop.

- **Menu**

Figure 4.5 Menu UI

Simple menu design with a picture of tree in the background

- **Player UI**

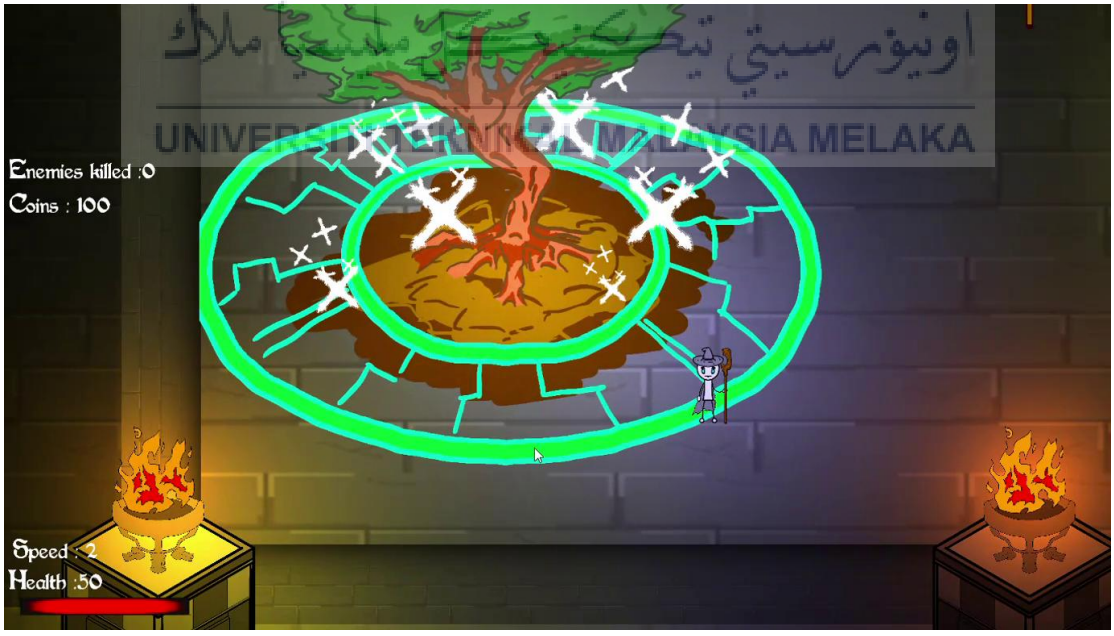


Figure 4.6 Player UI

Player user interface is displayed on the left side

- **Player controls**

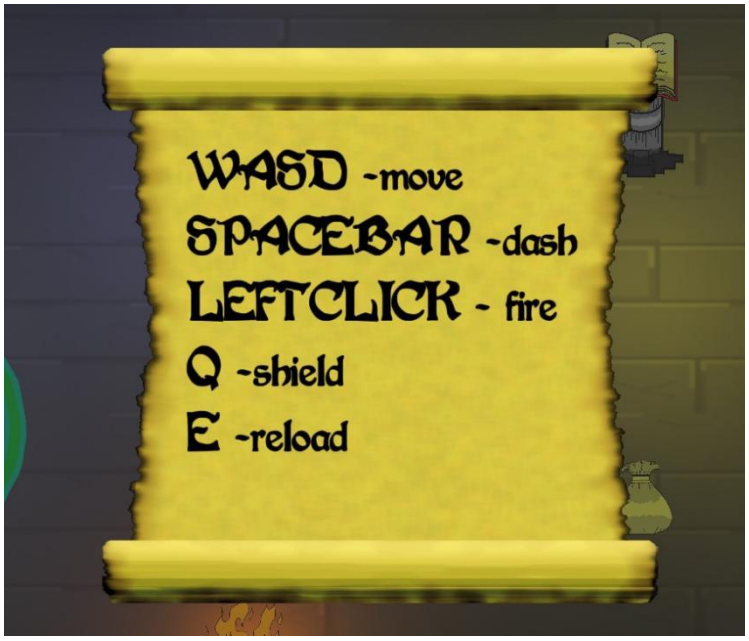


Figure 4.7 Player controls setting and abilities

Player can see the controls of the character and what key to use.

- **Pause Menu UI**

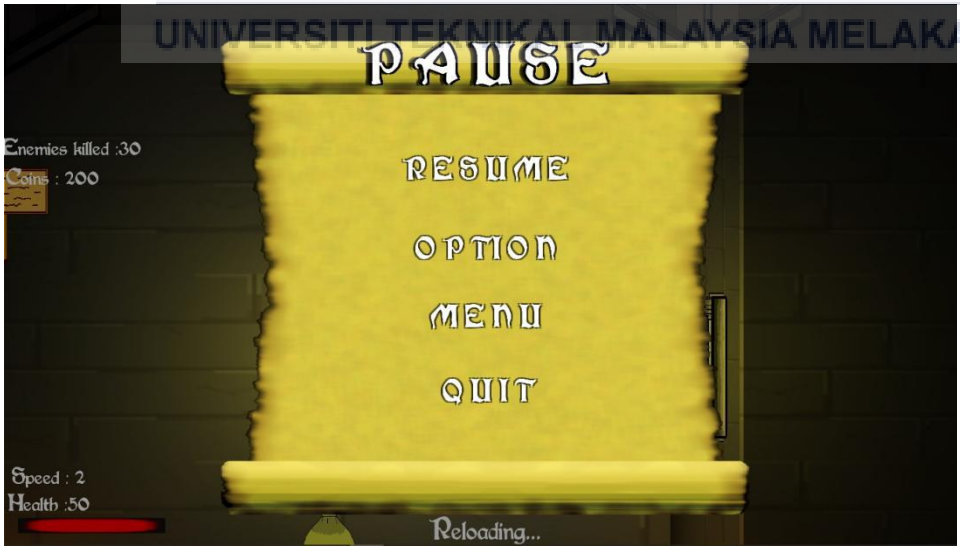


Figure 4.8 Pause menu UI

Everytime player press escape (ESC) it will activate the pause menu. The scene in the background is stopped for moment.

- **Shop UI**

Figure 4.9 Shop UI

The shop where player can use resources to upgrade player status.

- **Achievement UI**

Figure 4.10 Achievement UI

The list where player can see challenges and achievement.

- **Player Win/Lose condition**

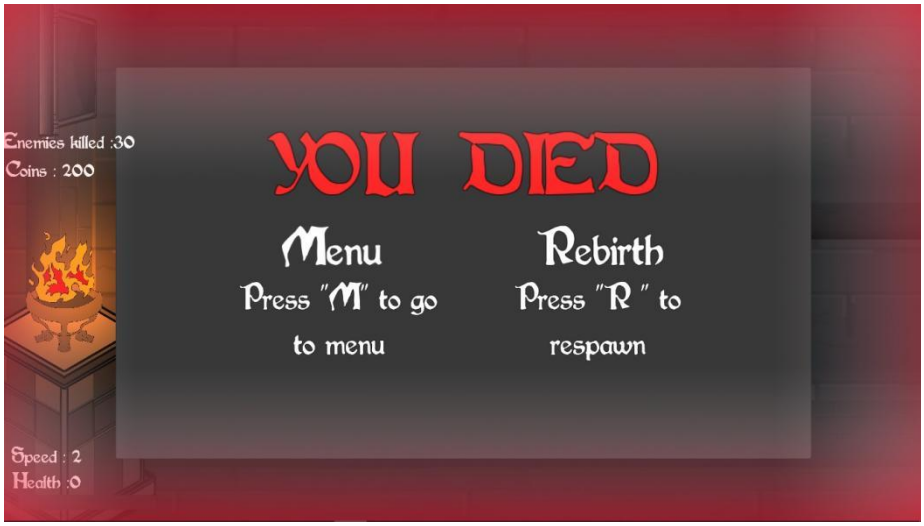


Figure 4.11 Player Die Panel

When player die the game will activate the panel to tell what player what to press to continue playing.

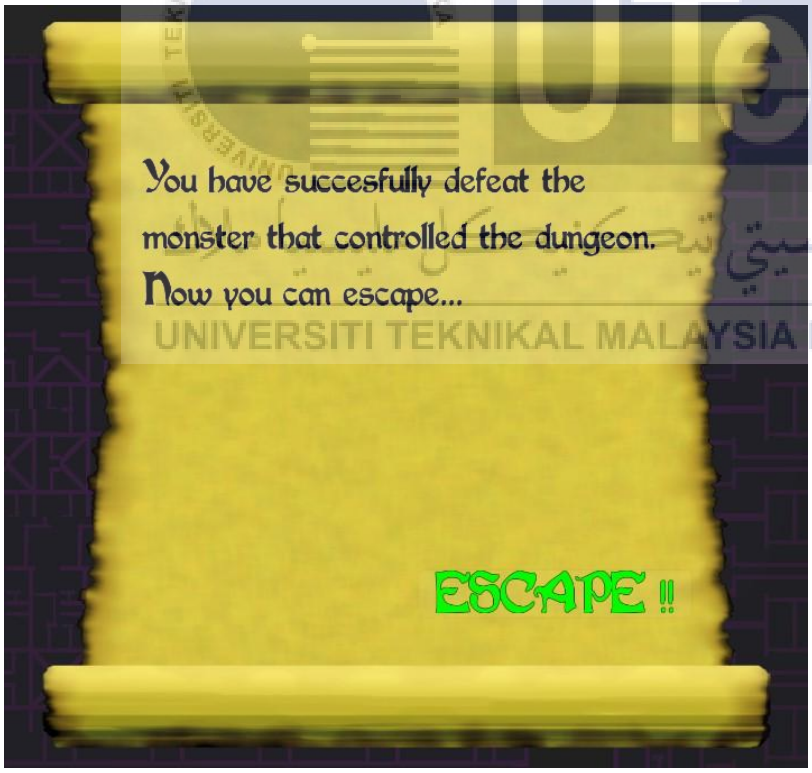


Figure 4.12 Player Escape Panel

When player successfully defeat the final boss, player can escape from the dungeon.

4. 4. Game Art

- **Game world**

  The game world is set in a fantasy-based dungeon. Player have to travel through three regions of the dungeon and defeat the final boss.

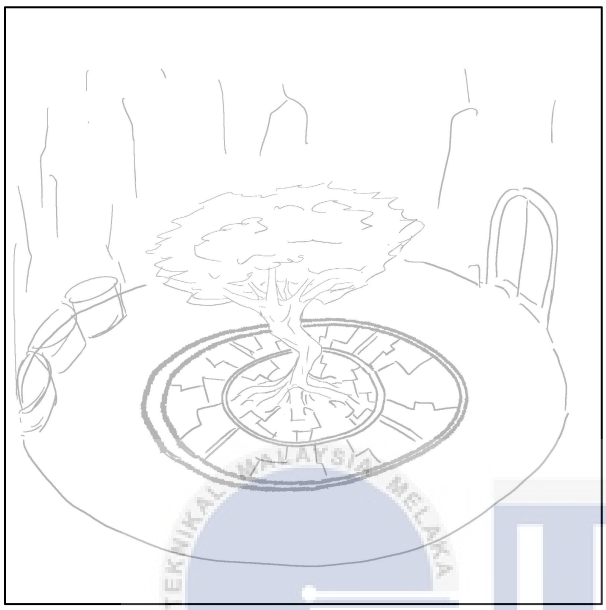


Figure 4.13 Sketch of starting scene

Figure 4.14 Project starting scene

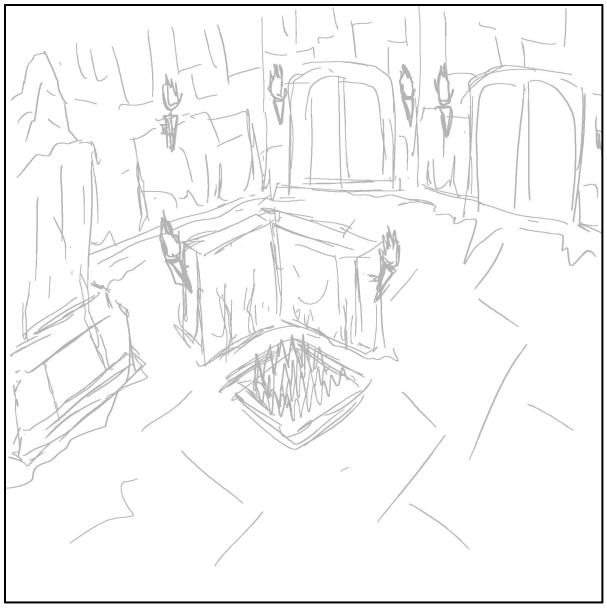


Figure 4.15 Sketch of level 1 environments

The first level is in ruin-like or old dungeon. The building and wall is man-made from abandoned ancient civilization. Now the place is in darkness, occupied by monsters.

Figure 4.16 Project level 1 environments

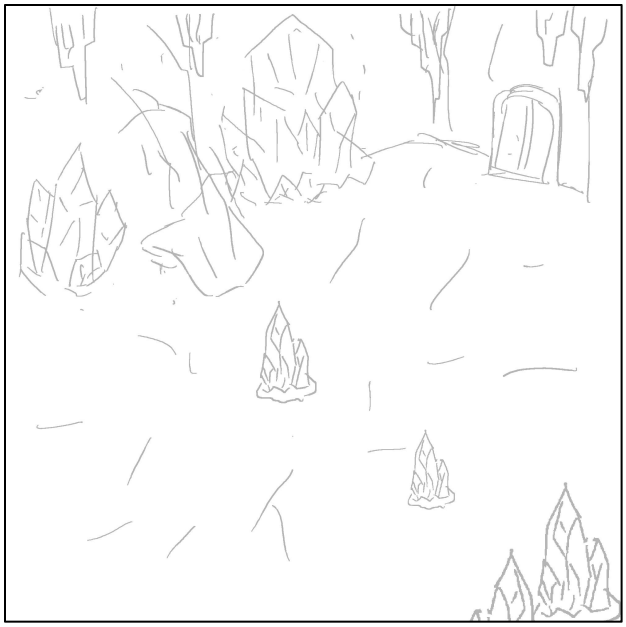


Figure 4.17 Sketch of level 2 environments

The second level is a crystal cave. Located right below the dungeon. This place used to be a crystal mining location in ancient civilization.

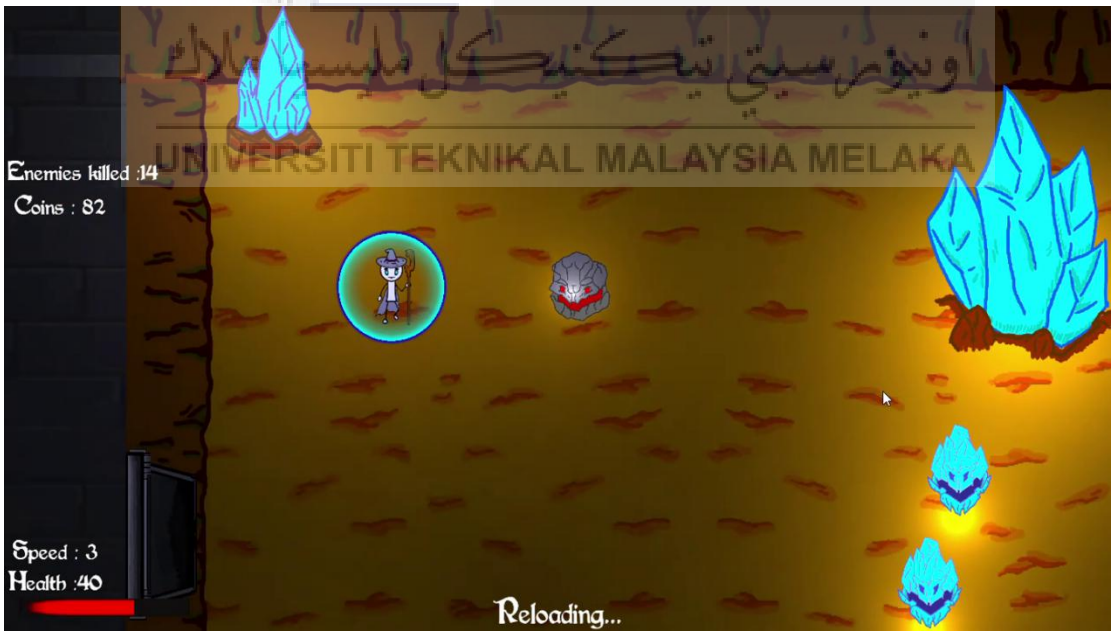


Figure 4.18 Project level 2 environments

Figure 4.19 Sketch of level 3 environments

The third level is lava cave. The level is formed in volcanic rock which is naturally formed by volcanic process from ancient ages. The lava is hot and very dangerous place to wander around.

Figure 4.20 project level 3 environments

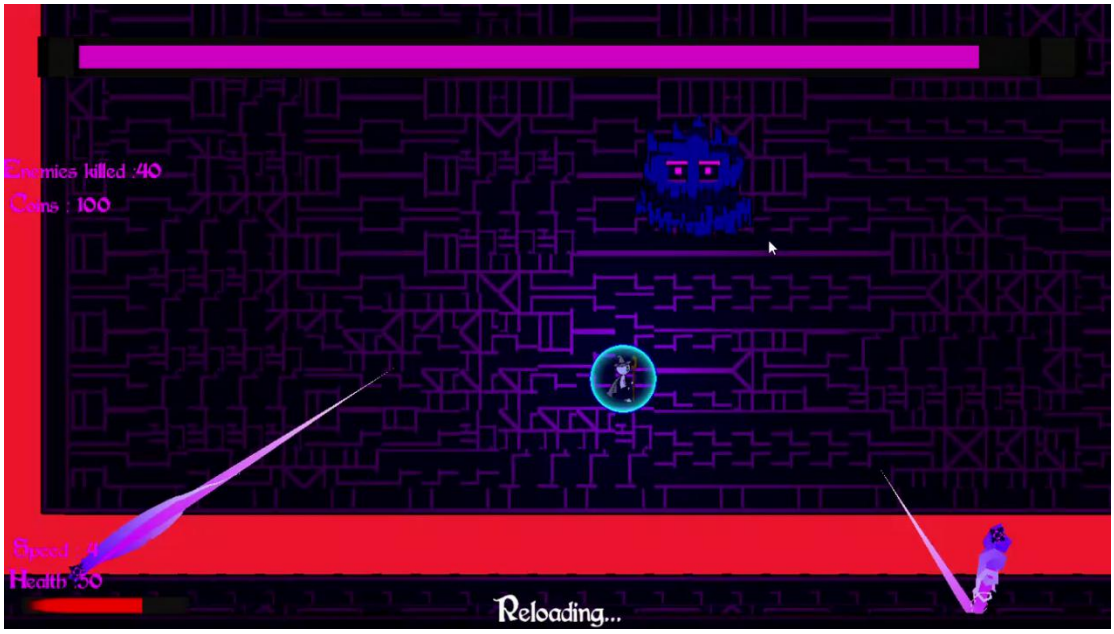


Figure 4.21 Final boss environment

The unknown place called "The End". The surrounding is from special material from ancient civilization and the room seems like it sealing something dangerous.

- **Character design**

  a. Main character

Figure 4.22 Main character asset

b. Enemies

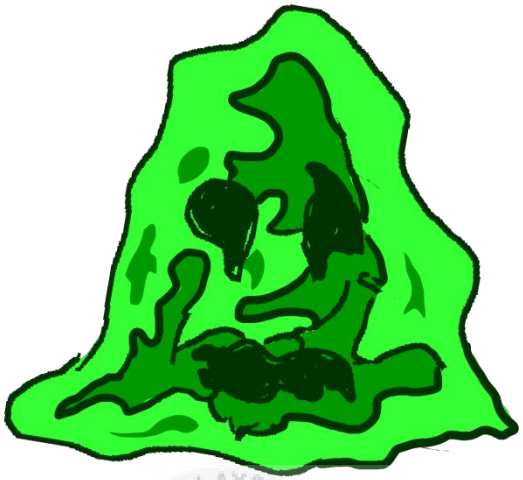


Figure 4.23 Slime

Figure 4.24 Rock golem

Figure 4.25 Crystal golem

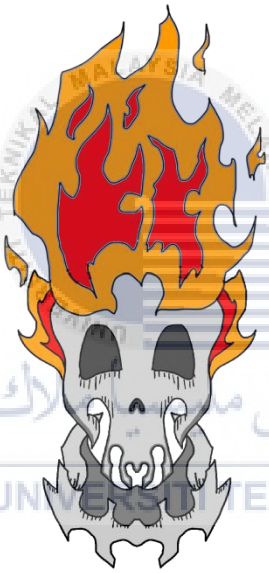


Figure 4.26 Burning Skull

Figure 4.27 Prime Slime

Figure 4.28 Crystal King

Figure 4.29Volcanic Demon

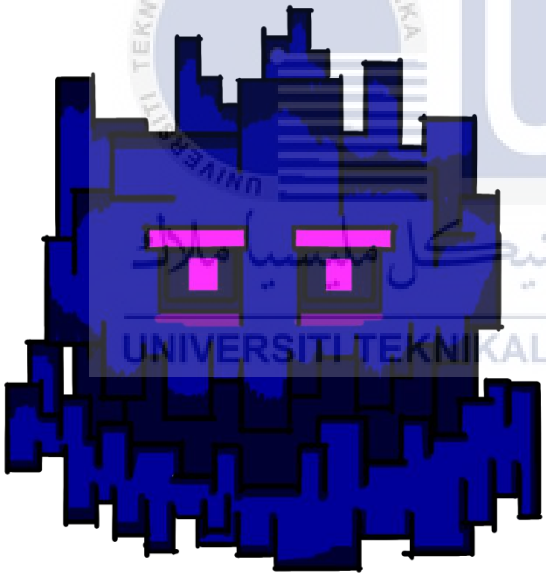


Figure 4.30 The Void

- **Camera model**

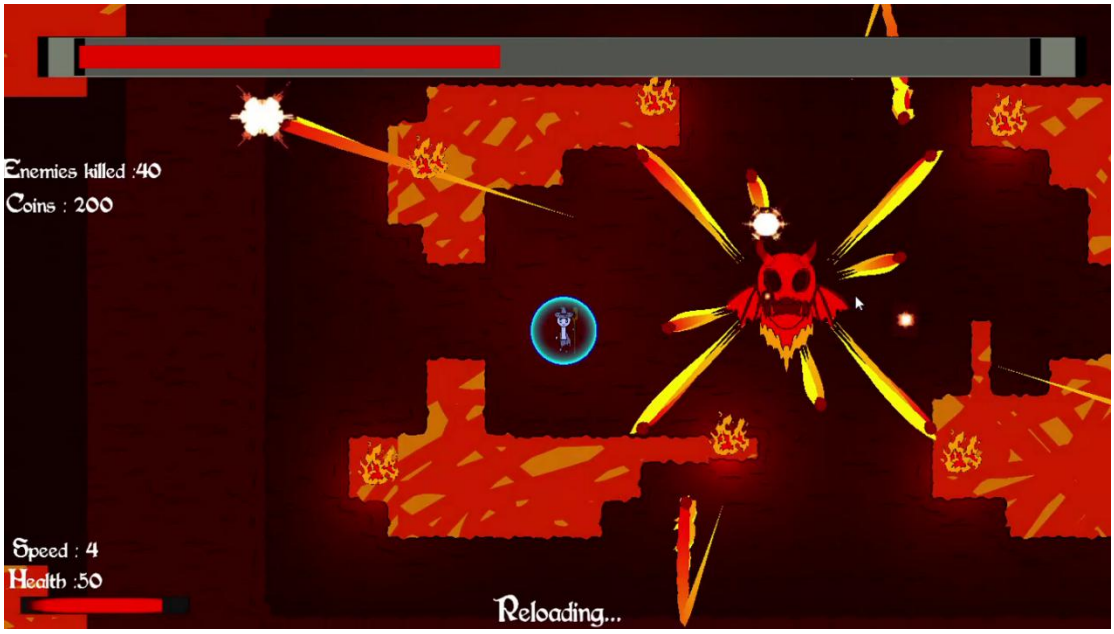


Figure 4.31 Orthographic camera view

- **Audio/Sound effect**

i. Environment Sound

ii. Combat sound effects

iii. Menu UI sound effects

4. 5. Conclusion

In conclusion, game design are really important in game development since it was the early phase of development that will convey concept before the implementations. In the next chapter, we will talk about the implementation of the design in the game.

CHAPTER 5

IMPLEMENTATION

5.1. Introduction

In this chapter we will highlight one of the important phase in game development which are the implementation. This chapter will provide information from the technical view of the implementation from programmer perspective during production of graphics, production of audio, production of video and production of animation. It also touch a little bit on the integration for the feature core mechanics in the game, configuration management and status

5. 2. Creation of Game Art

5. 2. 1. Production of Graphics

The production of game assets is created in Adobe Photoshop. Starts with sketches of characters and environments, continued with outlining and colouring of the assets. Then with Adobe Photoshop, it can generate a sprite sheets for assets animations.

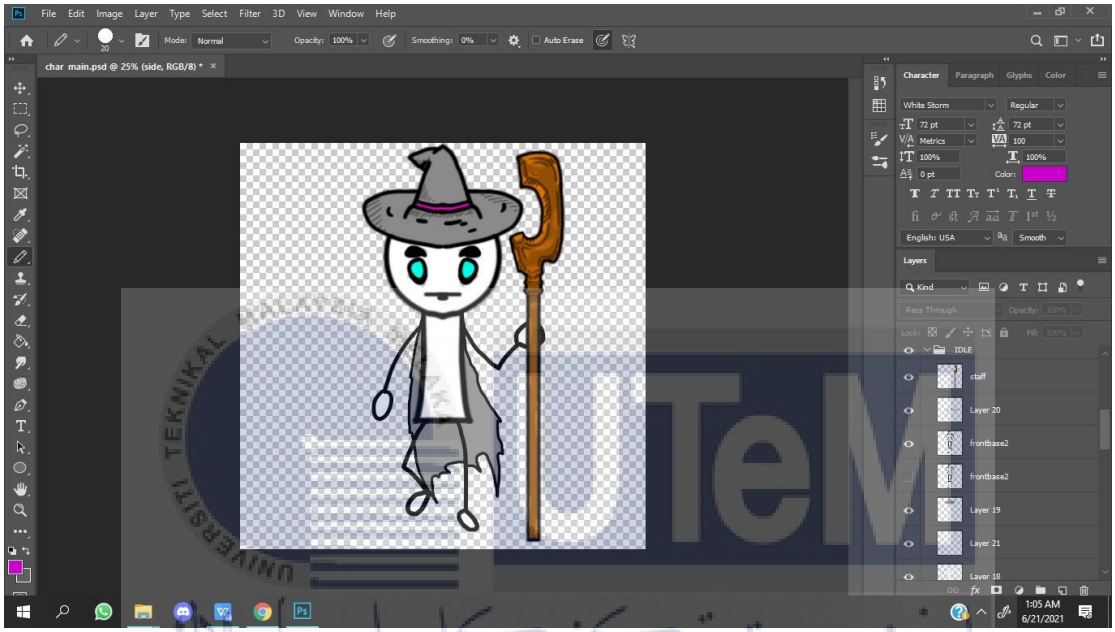


Figure 5.1 Character sprite creation

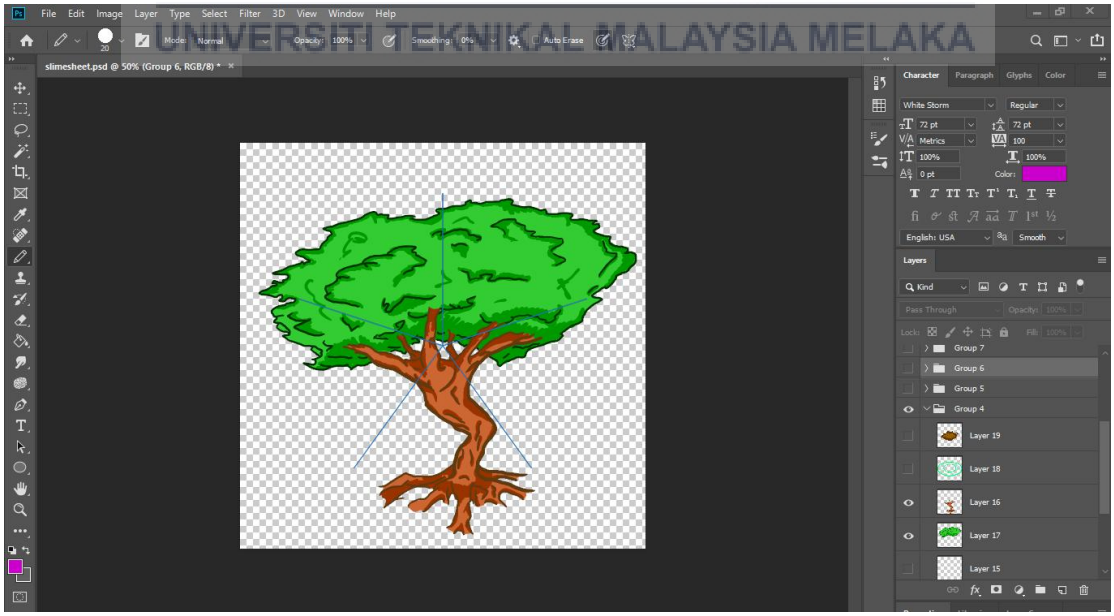


Figure 5.2 Environment sprite creation

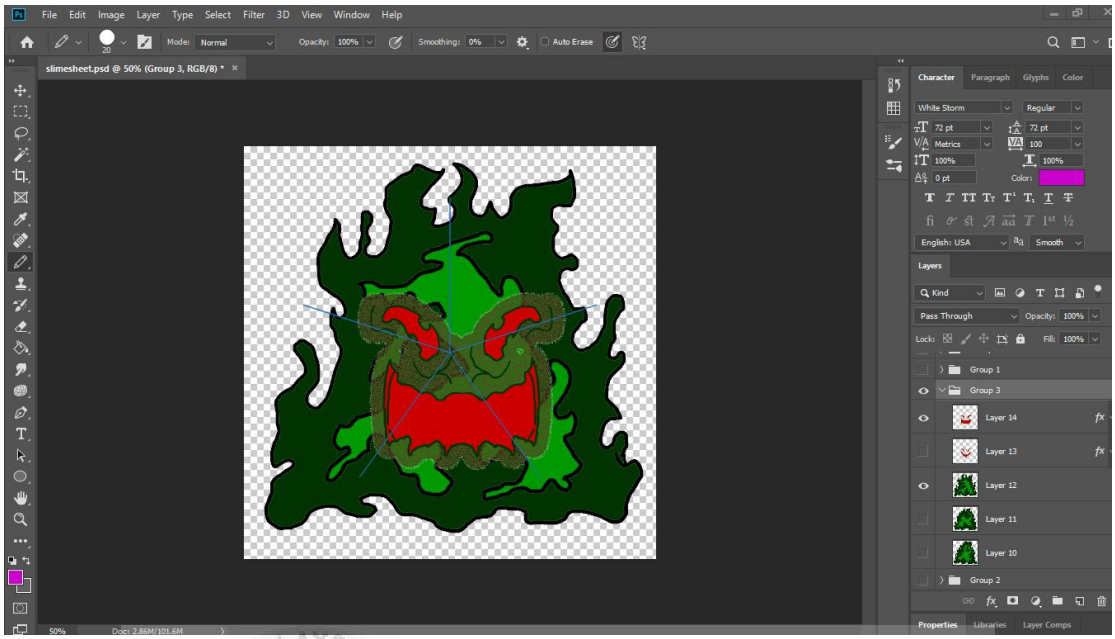


Figure 5.3 Enemy sprite creation

5. 2. 2. Production of Audio

Sound	Description
Ambient sound	Ambient sound that fit for environments like in a cave or ruins.
User Interface sound	Sound effect for menu and UI navigation. Hovering and clicking buttons.
Combat sound	Sound of player and enemies shooting projectiles.
Upgrade sound effects	Sound of player buy upgrades from shops.

Table 5.1 Audios in game

5. 2. 3. Production of Video

There is no video implementation inside the game.

5. 2. 4. Production of Animation

Figure 5.4 Player idle facing right sprite sheet

Figure 5.5 Player idle facing front sprite sheet

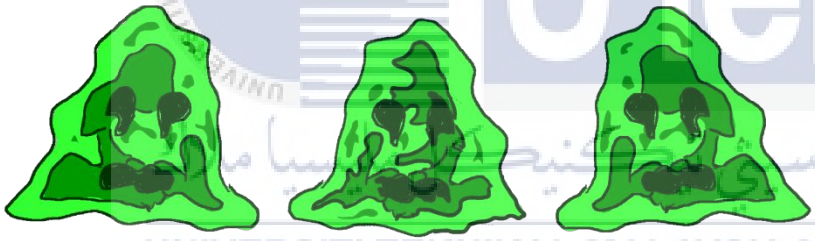


Figure 5.6 Enemy sprite sheet

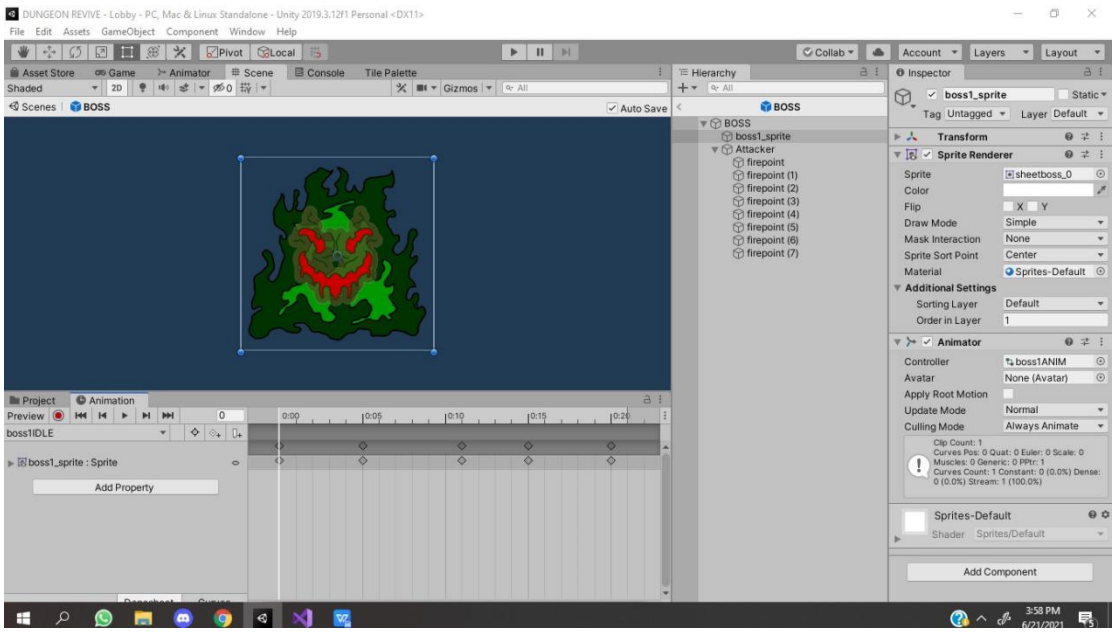


Figure 5.7 Enemy sprite animator

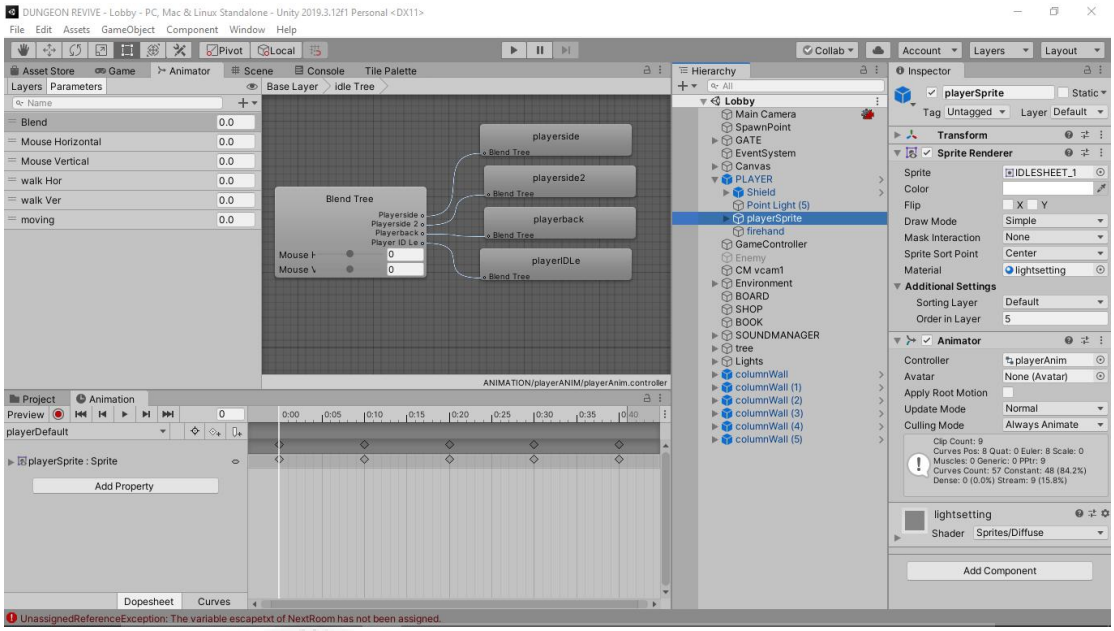


Figure 5.8 Character blend tree

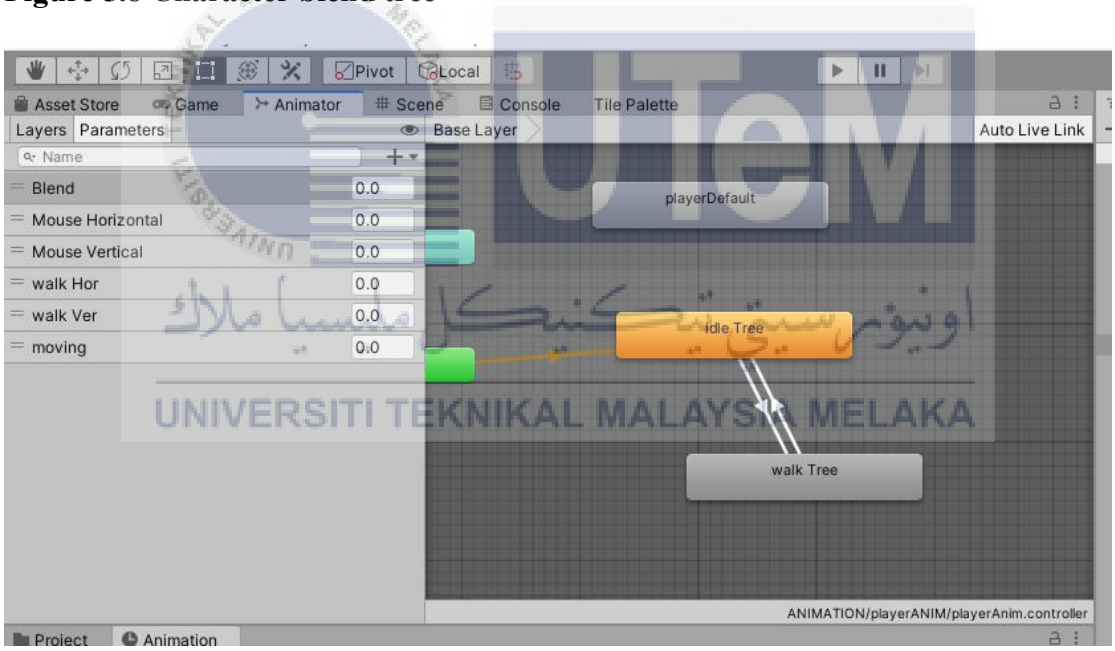


Figure 5.9 Animation transition

Some of the sprites is animated using sprite sheets and some which has separate parts is animated using Unity animator which using keyframes, nodes and blend trees.

5. 3. Integration of Game Components

- **Player movement**

Source code for player movement and player dashing. For movement there is variable called animator for controlling player animations.

```
speed = GameController.MoveSpeed;
movement.x = Input.GetAxis("Horizontal");
movement.y = Input.GetAxis("Vertical");
moveDir = new Vector2(movement.x, movement.y).normalized;
mousePos = cam.ScreenToWorldPoint(Input.mousePosition);

posDif = mousePos - rb.position;
animator.SetFloat("Mouse Horizontal", posDif.x);
animator.SetFloat("Mouse Vertical", posDif.y);
animator.SetFloat("walk Hor", movement.x);
animator.SetFloat("walk Ver", movement.y);
animator.SetFloat("moving", moveDir.magnitude);


//animator.SetBool("isWalk", true);

if (Input.GetKey(KeyCode.Space) && canDash == true)
{
    rb.MovePosition(transform.position + (transform.forward * (speed * Time.deltaTime)));
    rb.AddForce(movement.normalized * 6000);
    canDash = false;
    dashCD = 80;
```

Figure 5.10 player movement script

- **Player combat mechanics**

Script for player combat mechanic. Buy uprade from the shop to improve the combat mechanics.

```
if (Input.GetButtonDown("Fire1") && canShoot == true)
{
    Shoot();
    shootCount = shootCount + 1;

    if (shootCount == 5)
    {
        canShoot = false;
        shootCD = 50;
        StartCoroutine(shootCountReset());
    }
}

if (UnlimitedShot==true)
{
    unlimitedshoot = true;
}


if (unlimitedshoot==true)
{
    if (Input.GetButton("Fire1") )
    {
        Shoot();

    }
}
```

Figure 5.11 player shooting mechanics

```
if (Input.GetKey(KeyCode.E))
{
    StartCoroutine(shootCountReset());
}

if (Input.GetKey(KeyCode.Q))
{
    if (alreadySpawned==false)
    {
        spawnShield();
    }

}
```

Figure 5.12 player defence and reloading mechanics

- **Game controller script**

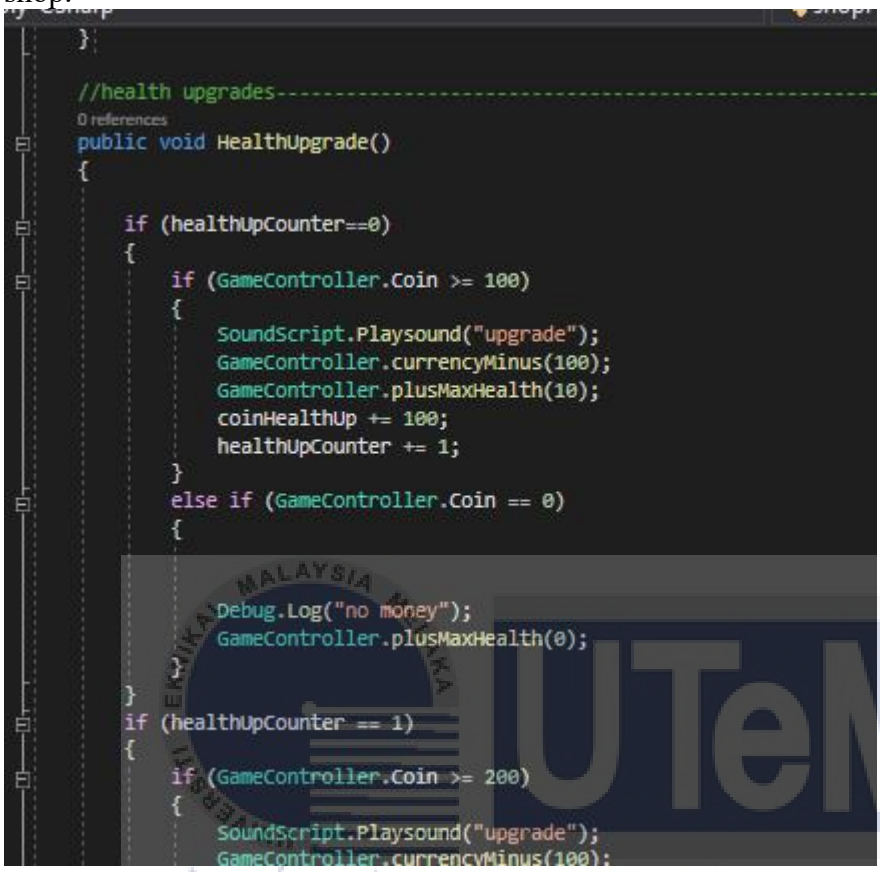
All the datas and conditions is controlled by game controller, player health data, resources data, movement speed data , player win and lose condition and etc.

```
void Update()
{
    healthText.text = "Health :" + health;
    coinText.text = "Coins : " + coin;
    speedText.text = "Speed : " + moveSpeed;
    enemykillText.text = "Enemies killed :" + enemyKilledCount;

    if (isPlayerDead==true)
    {
        //deathPanel.SetActive(true);
        Time.timeScale = 0f;
    }
    else
    {
        //deathPanel.SetActive(false);
        Time.timeScale = 1f;
    }

    if (Health <= 0)
    {
        isDead = true;
        PlayerDieScreenSpawn();
        player.SetActive(false);
    }
```

Figure 5.13 game controller script

- **Shop upgrades**

Player buy health, speed, bullet travel time and shoot mechanics upgrades from the shop.



```
    }

    //health upgrades-----------------------------------------------------
    0 references
    public void HealthUpgrade()
    {

        if (healthUpCounter==0)
        {
            if (GameController.Coin >= 100)
            {
                SoundScript.Playsound("upgrade");
                GameController.currencyMinus(100);
                GameController.plusMaxHealth(10);
                coinHealthUp += 100;
                healthUpCounter += 1;
            }
            else if (GameController.Coin == 0)
            {

                Debug.Log("no money");
                GameController.plusMaxHealth(0);
            }
        }
        if (healthUpCounter == 1)
        {
            if (GameController.Coin >= 200)
            {
                SoundScript.Playsound("upgrade");
                GameController.currencyMinus(100);
```

Figure 5.14 shop script

- **Achievements**

Variable is passed from game controller script to detect how many or what challenges that player have completed.

```
0 references
public void achieve1Complete()
{
    if (GameController.EnemyKilledCount >= 20 && b1disable==false)
    {
        GameController.currencyAdd(200);
        notcomplete1.SetActive(false);
        complete1.SetActive(true);
        b1disable = true;


    }
}

0 references
public void achieve2Complete()
{
    if (GameController.EnemyKilledCount >= 40 && b2disable == false)
    {
        GameController.currencyAdd(300);
        notcomplete2.SetActive(false);
        complete2.SetActive(true);
        b2disable = true;
```

Figure 5.15 achievements script

- **Enemy AI**

All the functionality and how the enemy acting is coded in the script.

```
1 reference
public void Attack()
{
    if (!cdAttack)
    {
        switch (enemyType)
        {
            case (EnemyType.Melee):
                GameController.DamagePlayer(10);
                StartCoroutine(Cooldown());
                break;
            case (EnemyType.Ranged):
                SoundScript.Playsound("enemyfire");
                GameObject bullet = Instantiate(bulletPrefab, firePoint.position, Quaternion.identity) as GameObject;
                bullet.GetComponent<enemybulletController>().GetPlayer(player.transform);
                bullet.GetComponent<Rigidbody2D>().gravityScale = 0f;
                bullet.GetComponent<enemybulletController>().isEnemyBullet = true;
                //bullet.GetComponent<bulletController>().speed = your_desired_speedf;
                StartCoroutine(Cooldown());
                break;
        }
    }
}
```

Figure 5.16 enemy AI script

- **Boss AI**

Boss attack pattern and animator is called using the script.

```
public void Attack()
{
    if (!cdAttack)
    {
        SoundScript.Playsound("enemyfire");
        GameObject bullet = Instantiate(bulletPrefab, firePoint.position, firePoint.rotation);
        Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();
        bullet.GetComponent<enemybulletController>().isEnemyBullet = true;
        rb.AddForce(firePoint.up * bulletForce, ForceMode2D.Impulse);
        Destroy(bullet, 2f);

        GameObject bullet2 = Instantiate(bulletPrefab, firePoint2.position, firePoint2.rotation);
        Rigidbody2D rb2 = bullet2.GetComponent<Rigidbody2D>();
        bullet2.GetComponent<enemybulletController>().isEnemyBullet = true;
        rb2.AddForce(firePoint2.up * bulletForce, ForceMode2D.Impulse);
        Destroy(bullet2, 2f);

        GameObject bullet3 = Instantiate(bulletPrefab, firePoint3.position, firePoint3.rotation);
        Rigidbody2D rb3 = bullet3.GetComponent<Rigidbody2D>();
        bullet3.GetComponent<enemybulletController>().isEnemyBullet = true;
        rb3.AddForce(firePoint3.up * bulletForce, ForceMode2D.Impulse);
        Destroy(bullet3, 2f);

        GameObject bullet4 = Instantiate(bulletPrefab, firePoint4.position, firePoint4.rotation);
        Rigidbody2D rb4 = bullet4.GetComponent<Rigidbody2D>();
        bullet4.GetComponent<enemybulletController>().isEnemyBullet = true;
        rb4.AddForce(firePoint4.up * bulletForce, ForceMode2D.Impulse);
```

Figure 5.17 Boss AI script

- **Bullet Controller**

Code to check what projectile hit

```
0 references
public void OnCollisionEnter2D(Collision2D collision)
{
    GameObject explodefx = Instantiate(hitfx, transform.position, Quaternion.identity);
    Destroy(explodefx, 0.1f);
    // Destroy(gameObject);
}

0 references
void OnTriggerEnter2D(Collider2D col)
{
    if (col.tag == "Enemy" && !isEnemyBullet)
    {
        col.gameObject.GetComponent<enemyController>().Death();
        Destroy(gameObject);
        GameObject explodefx = Instantiate(hitfx, transform.position, Quaternion.identity);
        Destroy(explodefx, 0.1f);
        //enemyController.DamageEnemy(5);
    }

    if (col.tag == "Player" && isEnemyBullet)
    {
        GameController.DamagePlayer(10);
        Destroy(gameObject);
    }
```

Figure 5.18 bullet controller script

- **Prefabs**

Prefab or a blueprint that spawned. Each have different animations or functionality.

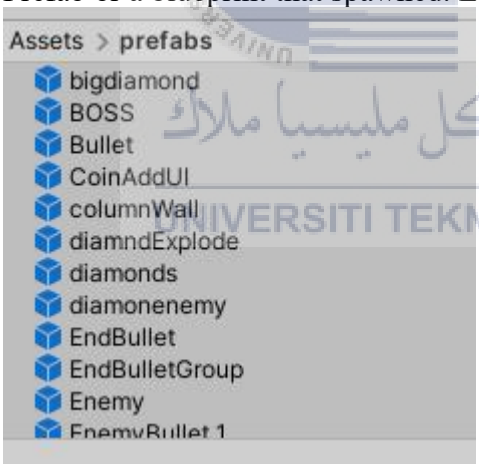


Figure 5.19 prefabs

5. 4. Game Configuration Management

The game can be play by two options which are through installation or click and play that require player to transfer the whole game file instead just small installation file. The game only need input devices like mouse and keyboard to play.

5. 4. 1. Configuration Setup

This project were published using Unity 2019.3.12f1 for Windows 64-bit. Its require few configuration inside project settings for example we need to setup the scene that supposed to be the first interaction to the player when interacting with the application, building lighting depending on the quality and settings up other crucial information for the game like the project title, logo and etc.

5. 4. 2. Version Control Procedure

Table 5.2 Testing Phases

Testing phase	**Description**	**Detail**
Alpha	The game be tested to find out what work and don't work including major bugs that affect the game in the major aspects In form of early prototype and must be at least playable. Mostly will be focusing on mechanics and localize content. This will be implement during Final Year Project 1	In form of early prototype and must be at least playable. Mostly will be focusing on mechanics and localize content. This will be implement during Final Year Project 1.
Beta	The game will be tested by target focus group where the feedback and questionaire will be collected from their perspectives for further improvement to polish the game.	After the evaluation from the evaluator and supervisor during Final Year Project 1 presentation. The game will enter semi Beta phase where the game will be improve in term of the crucial aspects such as the objective for the game itself. Then it will be enter the remaining phase during Final Year Project 2 where the feedback will be acquired from the target focus group.
Golden Version	The final version where the game is ready to publish to the market.	After evaluation in Final Year Project 2

5. 5. Implementation Status

Table 5.3 Implementation status

Component	Description	Duration To Complete	Completed Duration	Status
Game prototype creation	Basic mechanics of the game	2 Weeks	2 Weeks	On time
Game assets creation	Game assets such as 2D sprites and User Interface were created in CS Photoshop.	1 Week	1 Week	On time
Game World Creation	Creation of the game levels	1 Week	1 Week	On time
Interface Element and Implementation	Designing game user interfaces and user interfaces in the game engine using 2D elements	2 Weeks	10 days	In time
Game Mechanic Implementation	Implement all the functions and mechanics inside the game from the gameplay element until interface functionality	6 Weeks	6 Weeks	In time
Animation Implementation	The animation for the characters and game objects were implemented using Unity animator.	2 Weeks	2 Weeks	In time
Polishing the game	Fixing the bugs and make few adjustment before exporting the game.	2 Weeks	2 Weeks	In time

5. 6. Conclusion

Implementation phase are very important for the game mechanics to work and playable. Its include the creations of the game art which involves production of graphics, audio, video and animation. In term of game integration and C# language were use for this project by using Unity Engine to build the game. In the next chapter, testing for the target group user will be conduct for further feedback through questionaires.

CHAPTER 6

TESTING

6.1. Introduction

The next important phase after the implementation process is testing, which will be covered in this chapter. The goal of this project is to determine whether the game's replay values are effective based on the players' experience with the game.
The purpose of testing is not to find errors and bugs, but to ensure that the game met the project's objectives.A few number of tests is involved to meet the project functionality and usability.

6.2. Test Plan

After the Dungeon: Revive game has been developed, a test must be conducted to determine the system's functionality and usability. To ensure that tests run smoothly and produce the best results, preparation is required prior to conducting them. Identifying the test users, the test environment, the test schedule, the test strategy, the test design, and the test result and analysis are all part of the preparation.

6. 2. 1. Test Organization

The personnel involved in this chapter are game developer, the one who develop the game to do unit testing and while functionality testing with developers who have the experiences in game development. The next person is the end users to test the usability of the system.The end users are open to the public.

There are three types of test carried out, unit test, functionality test and usability test. Each tester has roles and responsibilities. The test organization are summarized in Table 6.1.

Table 6.1 Test Organization

Type of Test	Personnel Involved	Roles and Responsibilities
Unit	Game Developer (one person)	To test each software design component shows and works correctly.
Functionality	Game Developers (two persons)	To test each question whether it gets correct answer and functions well.
Usability	End Users (45 candidates)	To test the how players feel about playing the game.

6. 2. 2. Test Environment

The game will be tested in front of the general public from the perspectives of casual and hardcore gamers. However, due to a Movement Control Order (MCO) for Covid-19, testers will be given the option of either downloading the game or watching a developer-led video game demo to better understand the game's core concept. The testers were then asked to complete a series of questionnaires prepared by the developer.

I. Hardware Requirement

The testers must have a personal computer(PC) or a laptop to play the game. Depends on the testers availability to either play the game or just watch the video game demo to have better understanding of the game.

II. Software Requirement

The tester need to have a web browser to download the game files from Google Drive to test the game. As for the video game demo can be watched at Youtube page or Youtube application for mobile devices.

6. 2. 3. Test Schedule

Test schedule is made to ensure the tests run as planned within the period given. After that, the results will be analyzed to do improvements. Therefore, tests must hold in schedules so that the results can be presented on time.

6. 3. Test Implementation

There are several methods in conducting the usability testing. Only one method will be used during the usability testing which is by giving the game file download link, the video game demo link and the link for the online questionnaire that prepared by the developer of the project.

6. 4. Test Result and Analysis

An online survey by using Google Forms has been distributed to game testers. Google Forms is an online platform provided by Google to make online custom questionnaire with responses graphs. Total of 45 respondents has been giving feedbacks.

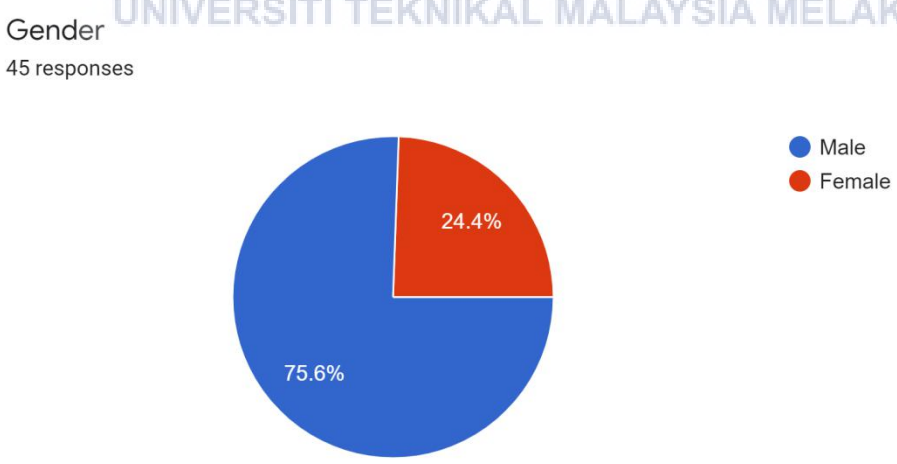


Figure 6.1 Gender data pie chart

Figure 6.1 shows that out of 45 respondents there are 34 males (75.6%) and 11 (24.4%) females that participated as end users test.

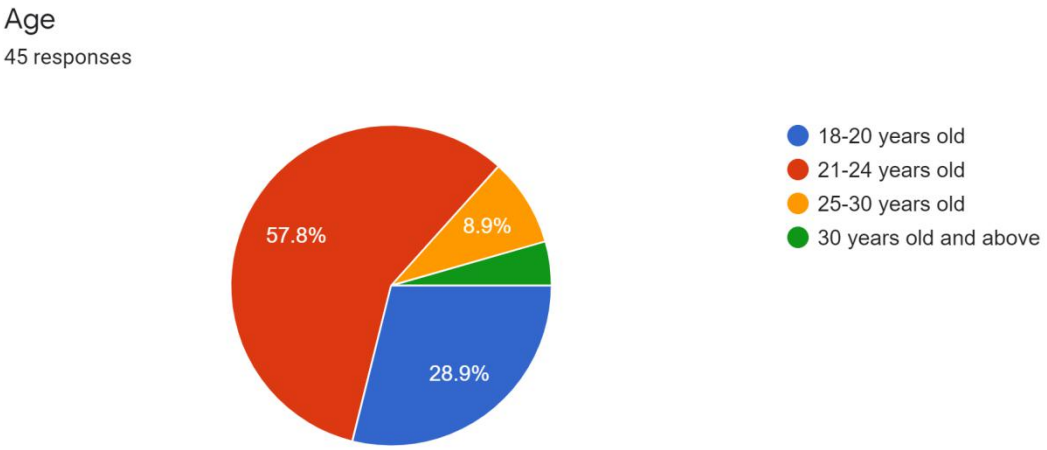


Figure 6.2 Age data pie chart

Figure 6.2 shows that out of 45 respondents there are 26 respondents (57.8%) from age 21 to 24 years old, 13 respondents(28.9%)from the age of 18 to 20 years old, 4 respondents (8.9%) from the age of 25 to 30 years old and 2 respondents (4.4%) from the age of 30 years old and above.

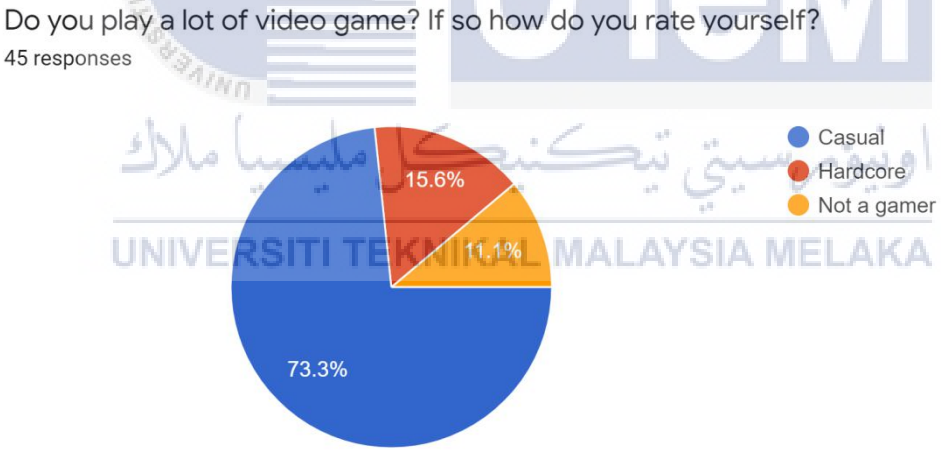


Figure 6.3 Gamer data pie chart

Figure 6.3 shows that out of 45 respondents there are 33 respondents (73.3%) that is a casual gamer, 7 respondents (15.6%) that is hardcore gamer and 5 respondents (11.1%) consider themself as not a gamer.

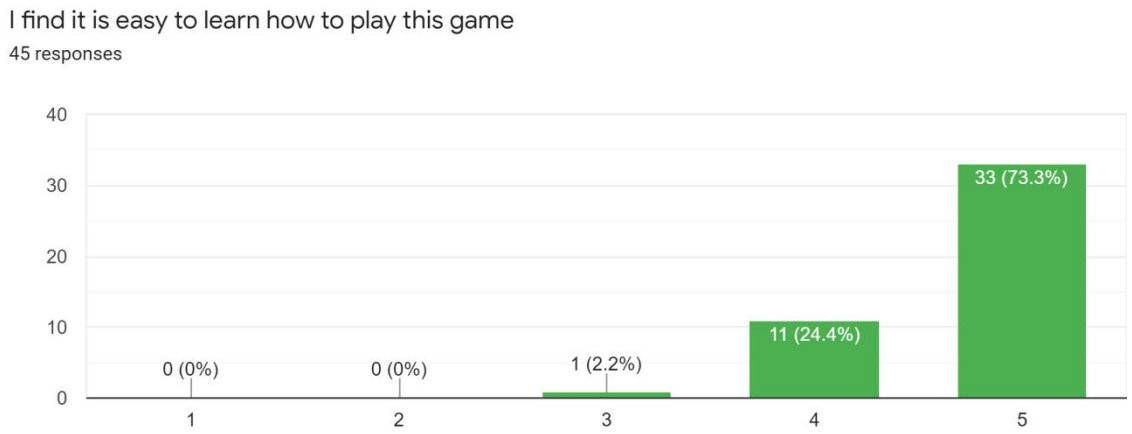


Figure 6.4 easy learn to play data

Figure 6.4 shows that out of 45 respondents how much they agree that the game is easy to learn how to play the game, there are 33 respondents (73.3%) that strongly agree , 11 respondents (24.4%) that is quite agree and 1 respondent (2.2%) that is neutral.

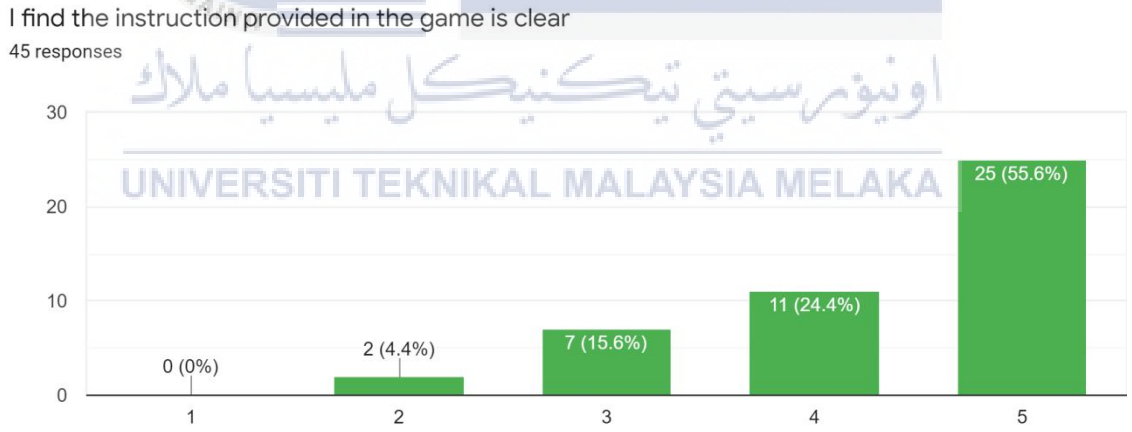


Figure 6.5 Instruction clear data

Figure 6.5 shows that out of 45 respondents how much they agree that the instruction in the game is clear, there are 25 respondents (55.6%) that strongly agree , 11 respondents (24.4%) that is quite agree, 7 respondent (15.6%) that is neutral and 2 respondents (4.4%) that is quite disagree.

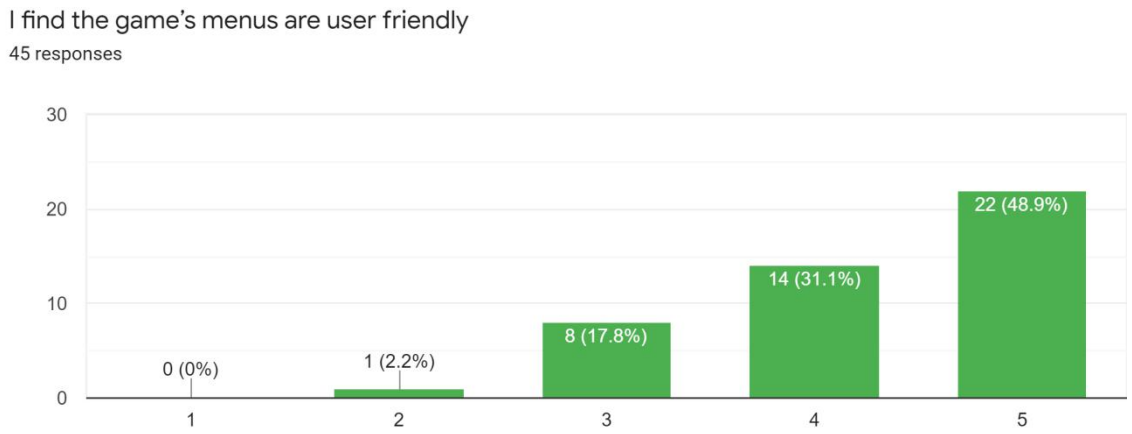


Figure 6.6 Game menus user friendly data

Figure 6.6 shows that out of 45 respondents how much they agree that the game's menus in the game are user friendly, there are 22 respondents (48.9%) that strongly agree , 14 respondents (31.1%) that is quite agree, 8respondent (17.8%) that is neutral and 1 respondents (2.2%) that is quite disagree.

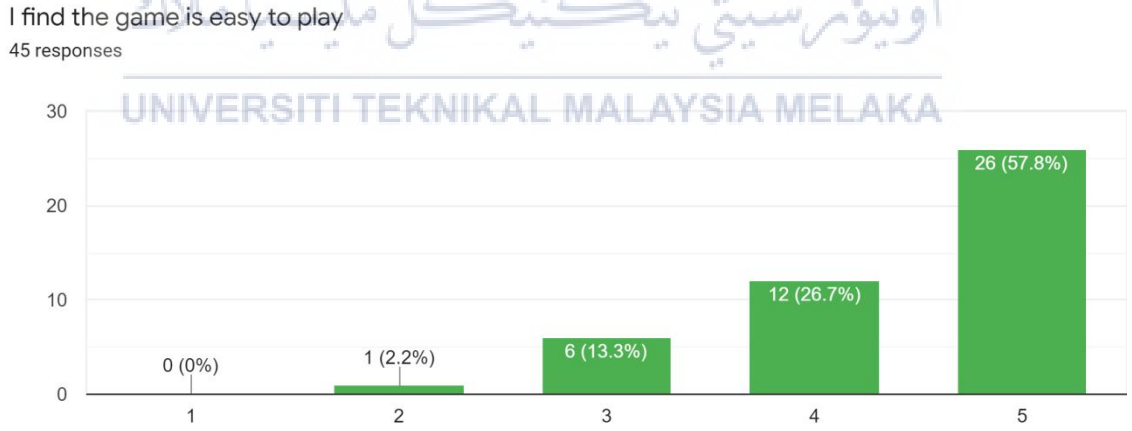


Figure 6.7 Easy to play data

Figure 6.7 shows that out of 45 respondents how much they agree that the game is easy to play, there are 26 respondents (57.8%) that strongly agree , 12 respondents (26.7%) that is quite agree, 6 respondent (13.3%) that is neutral and 1 respondents (2.2%) that is quite disagree.

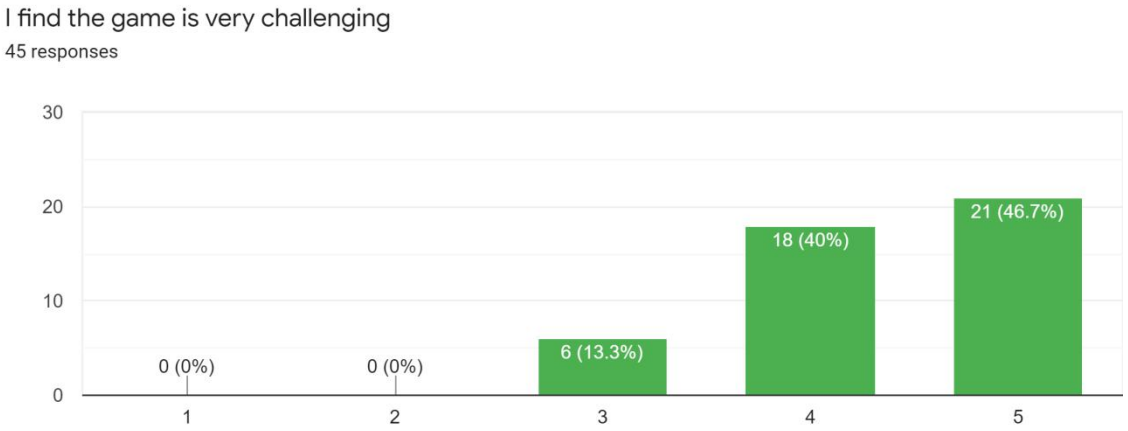


Figure 6.8 Challenging data

Figure 6.8 shows that out of 45 respondents how much they agree that the game is very challenging, there are 21 respondents (46.7%) that strongly agree , 18 respondents (40.0%) that is quite agree and 6 respondent (13.3%) that is neutral.

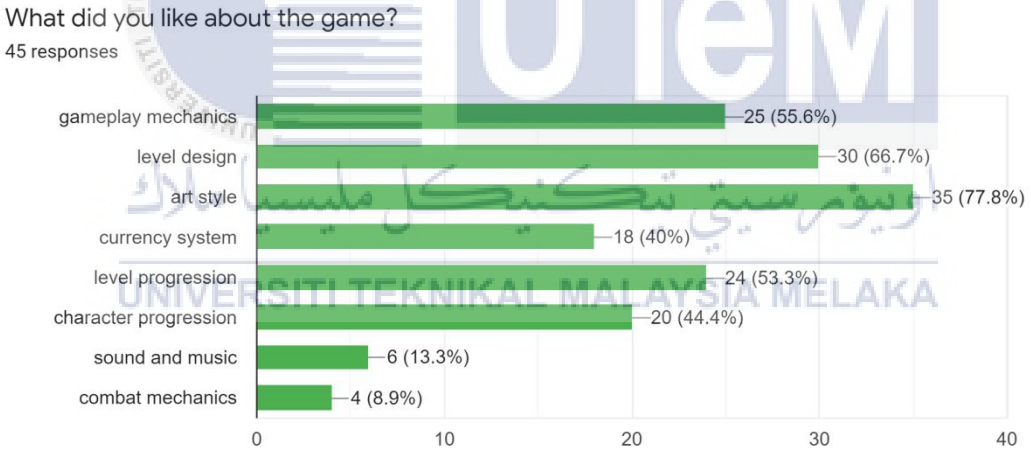
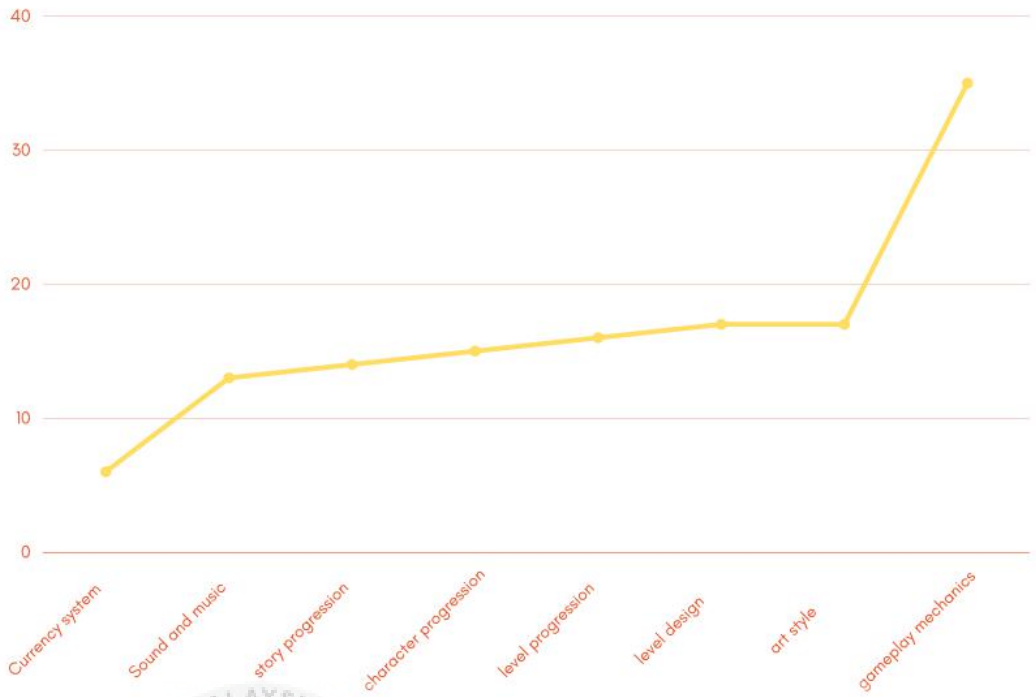


Figure 6.9 What liked about the game data



Data analysis from figure 6.13

Figure 6.9 shows that out of 45 respondents what did they like about the game, there are 25 respondents (55.6%) like the game mechanics, 30 respondents (66.7%) like the level design, 35 respondents (77.8%) like the art style, 18 respondents (40.0%) like the currency system, 24 respondents (53.3%) like the level progression, 20 respondents (44.4%) like the character progression, 6 respondents (13.3%) like the sound and music and 4 respondents (8.9%) like the combat mechanics.

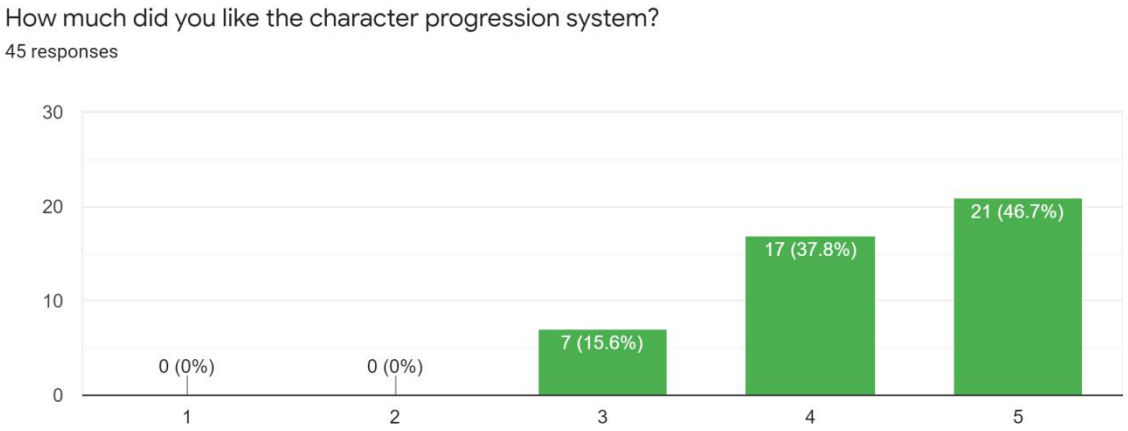


Figure 6.10 Character progression data

Figure 6.10 shows that out of 45 respondents how much they like the character progression system, there are 21 respondents (46.7%) that strongly agree , 17 respondents (37.8%) that is quite agree and 7 respondent (15.6%) that is neutral.

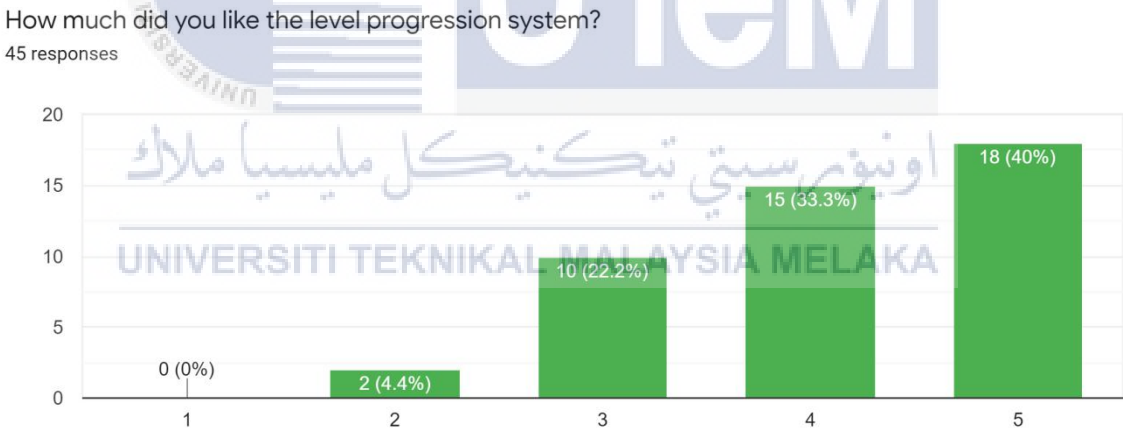


Figure 6.11 level progression data

Figure 6.11 shows that out of 45 respondents how much they like the level progression system, there are 18 respondents (40.0%) that strongly agree , 15 respondents (33.3%) that is quite agree, 10 respondent (22.2%) that is neutral and 2 respondents (4.4%) that is quite disagree.

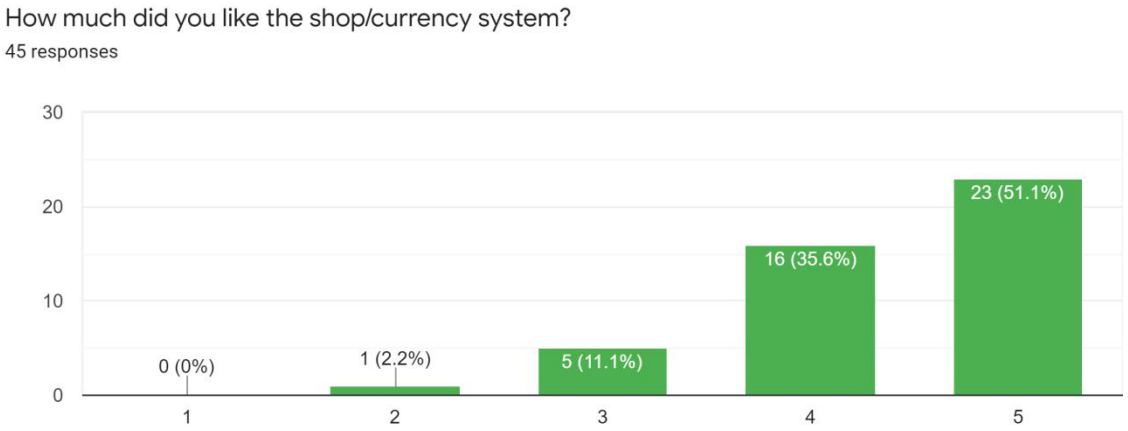


Figure 6.12 Shop/currency system data

Figure 6.12 shows that out of 45 respondents how much they like the shop/currency system, there are 23 respondents (51.1%) that strongly agree , 16 respondents (35.6%) that is quite agree, 5 respondent (11.1%) that is neutral and 1 respondents (2.2%) that is quite disagree.

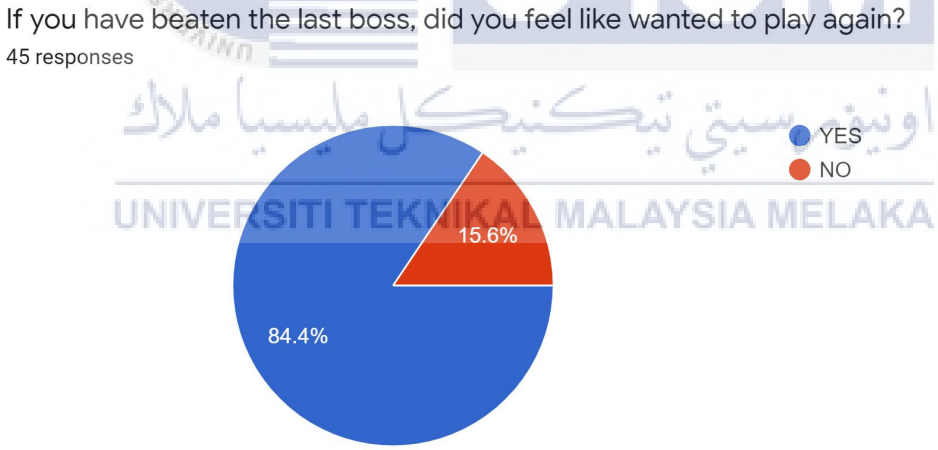
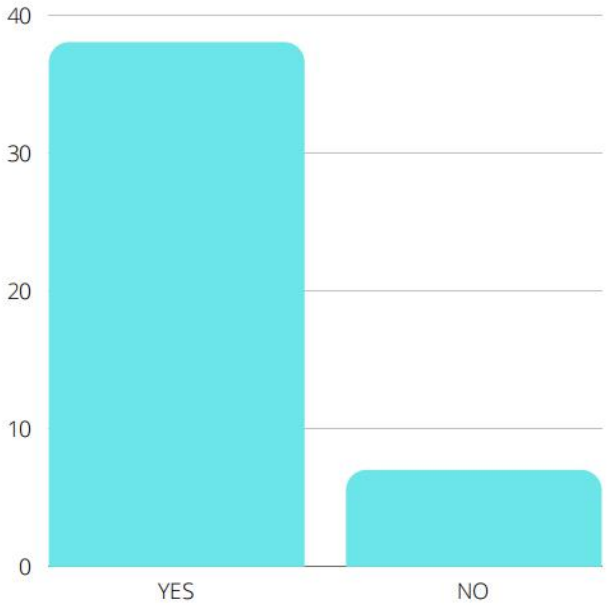


Figure 6.13 Play again data



Data analysis from figure 6.13

Figure 6.13 shows that out of 45 respondents if they play the game again after finishing the game for the first time, 38 respondents (84.4%) says Yes because they feel like wanted to explore the rest of the game and to get better result while 7 respondents(15.5%) says No because the mechanics feels clunky and the game overall is too hard .

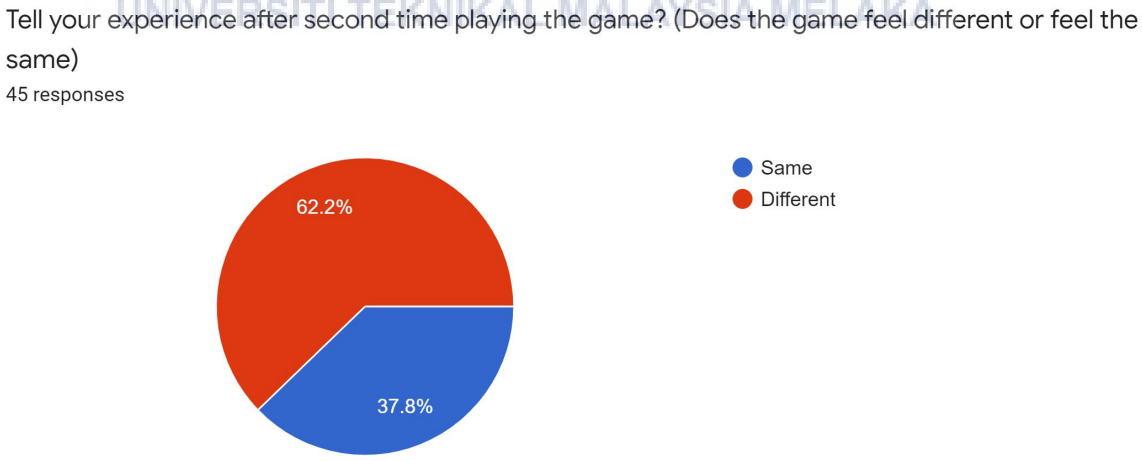
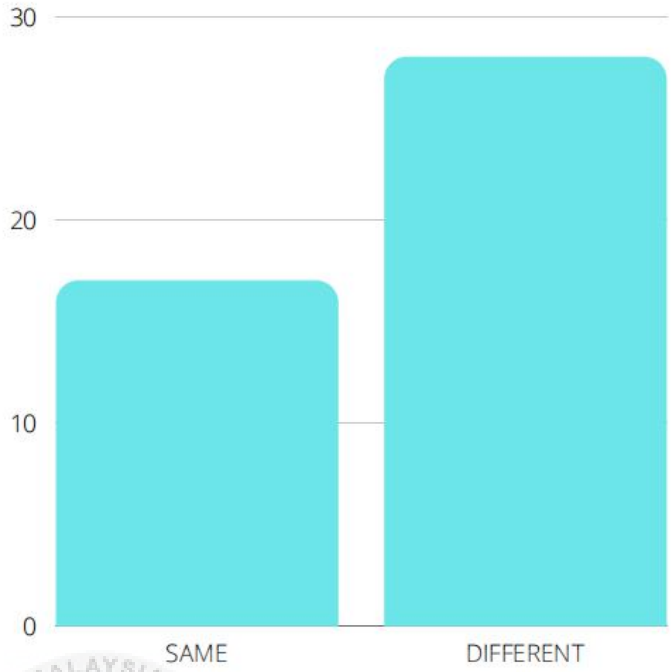


Figure 6.14 Experience second time data



Data analysis from figure 6.14

Figure 6.14 shows what 45 respondents the experience after the second time playing the game, 17 respondents (37.7%) say that the game feel the same but the enemies is stronger, 28 respondents (62.2%) say that the game feel different because their character is a lot stronger compared to when the started playing.

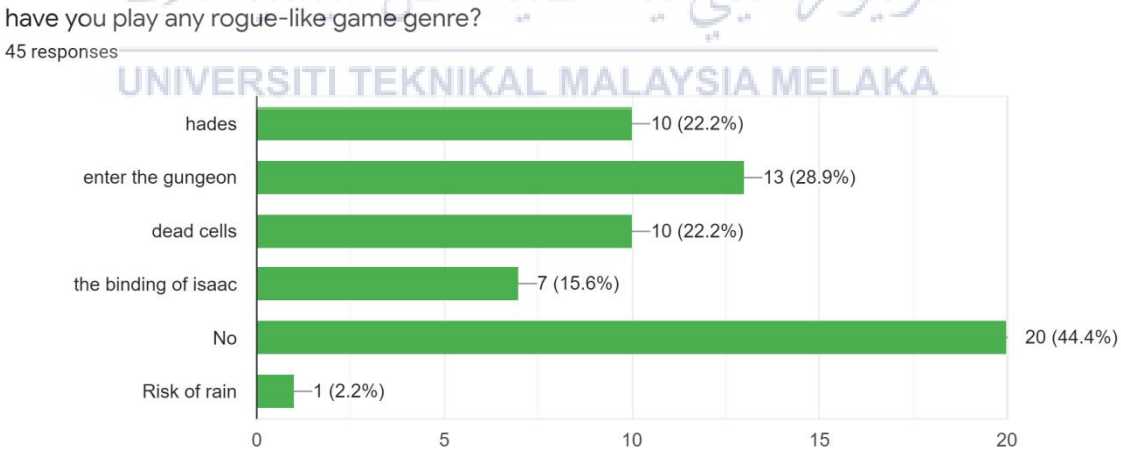
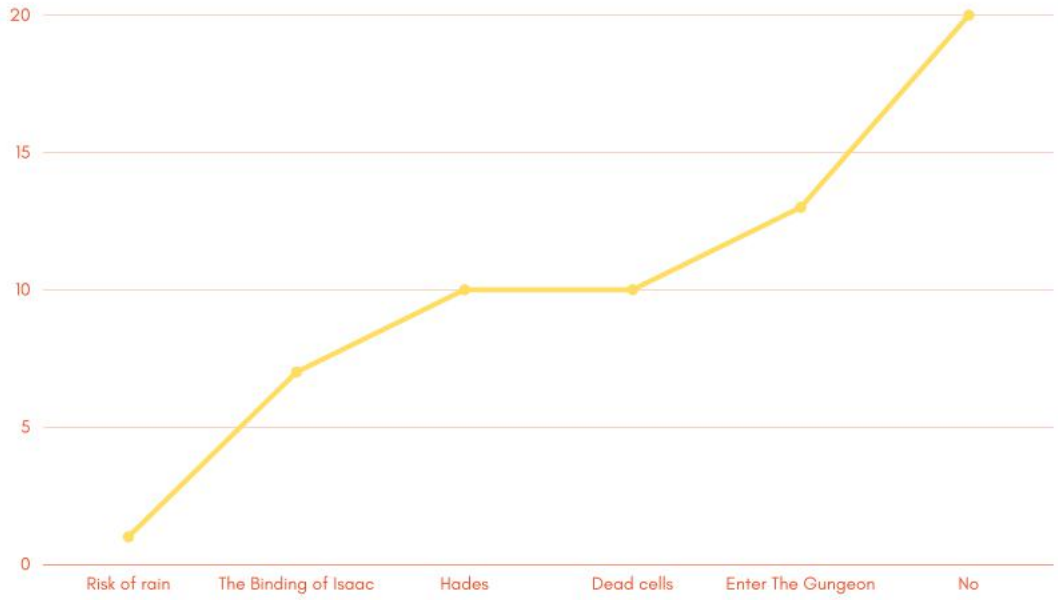


Figure 6.15 rogue-like game data



Data analysis from figure 6.15

Figure 6.15 shows that out of 45 respondents if they have ever play any rogue-like game genre, there are 10 respondents (22.2%) that have play Hades and Dead Cells , 13 respondents (28.9%) have play Enter the Gungeon , 7 respondent (15.6%) have play The Binding of Isaac, 1 respondent (2.2%) have play Risk of Rain and the majority of the respondents 20(44.4%) have never played any rogue-like game genre.

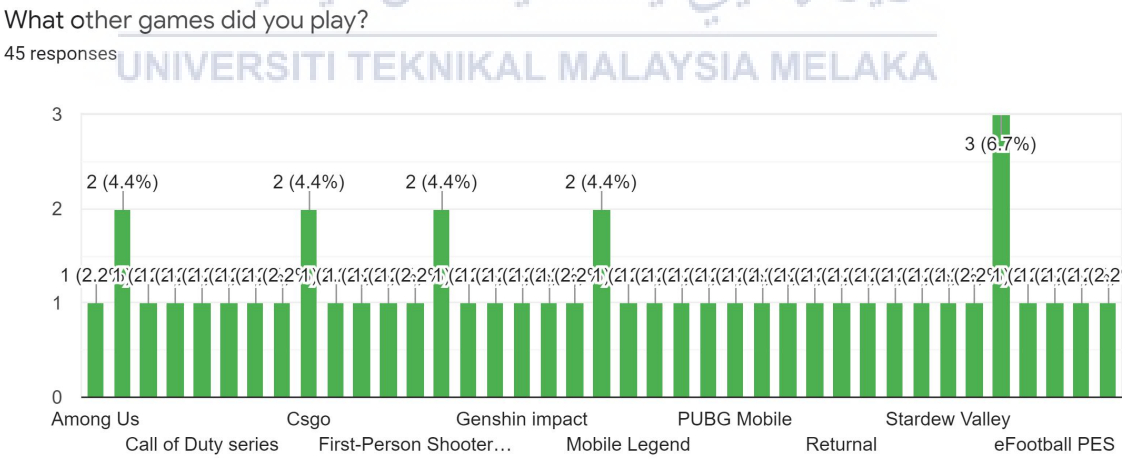


Figure 6.16 other game data

Genre	Number of respondents
Shooter Game	14
Indie games	12
Multiplayer games	7
MOBA games (multiplayer online battle arena)	6
RPG games	4
Sport games	2

Table 6.2 Count of other games played by respondents according to genre

Figure 6.16 shows that out of 45 respondents what other games that the respondents have played, this show the variety of games that the 45 respondents have played. The majority of games played is Valorant, Player Unknown Battleground and Minecraft for most played Indie game.

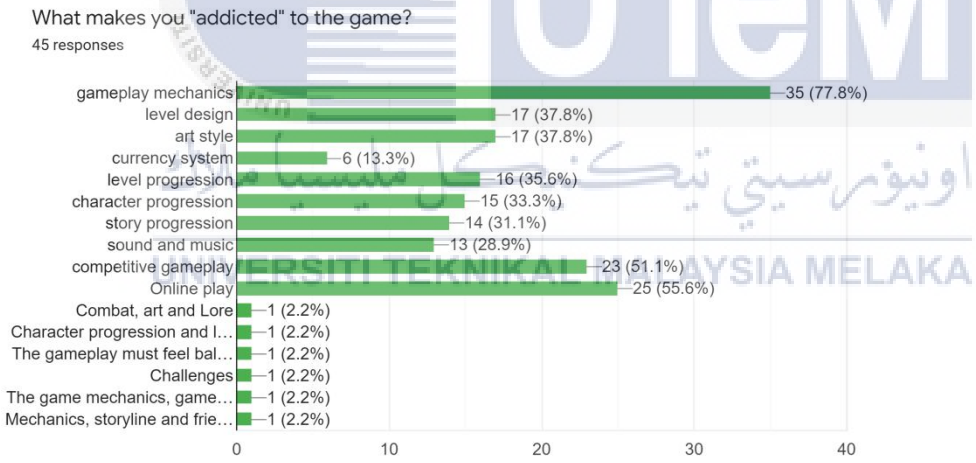


Figure 6.17 reason data

Figure 6.17 shows what makes the respondents keep playing the game the played, majority of the respondents play the game because of gameplay mechanics, online play and competitive gameplay. Level, character and story progression is the average for what makes people play the game.

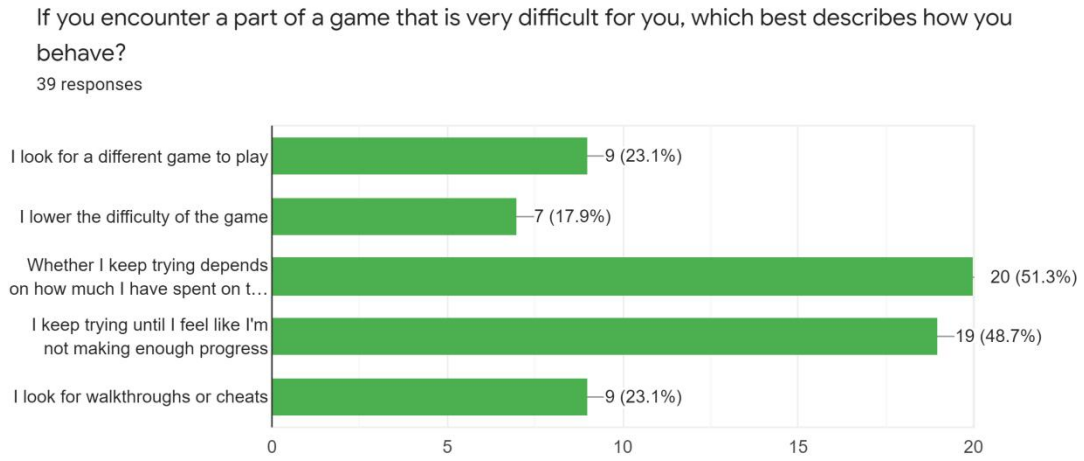
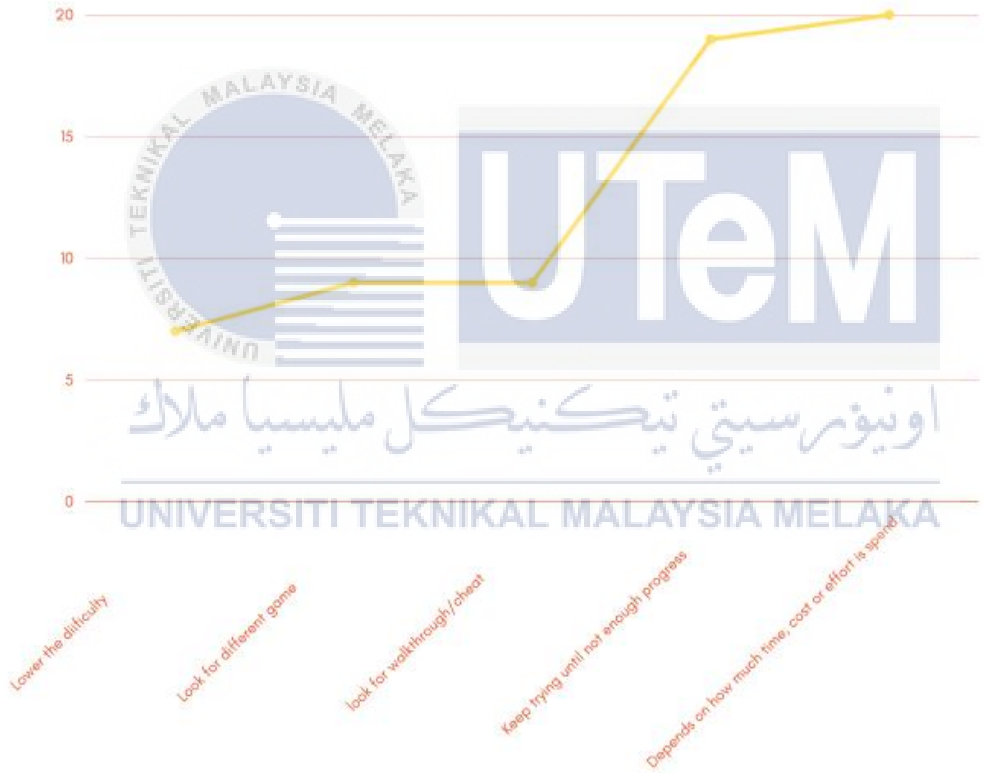


Figure 6.18 player behaviour data



Data analysis from figure 6.18

Figure 6.18 shows how respondents behave when the game is too difficult, 9 respondents (23.1%) look for a different game to play , 7 respondents (17.9%)lower the difficulty of the game , 20 respondent (51.3%) keep trying depends on how much I have spent on the game (time, money and effort) ,19 respondents (48.7%) keep trying until I feel like I'm not making enough progress and 9 respondents (23.1%) look for walkthrough or cheats.

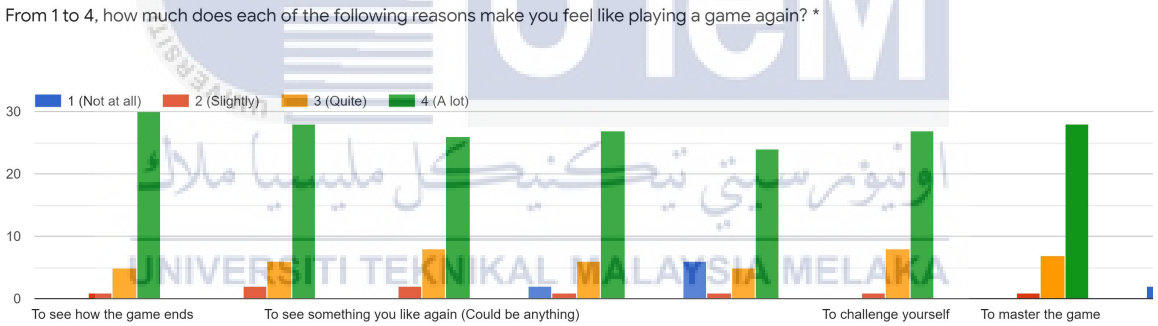


Figure 6.19 Reasons data

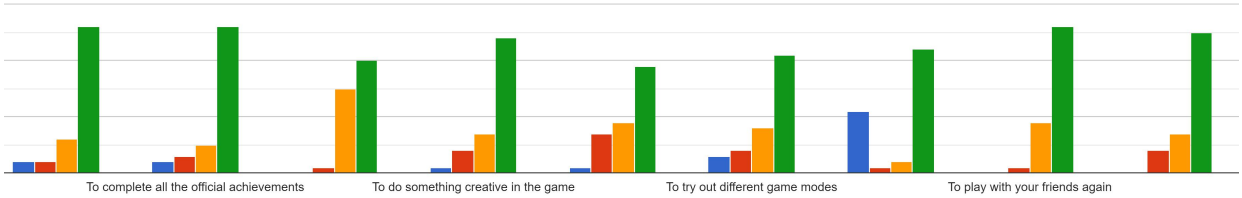
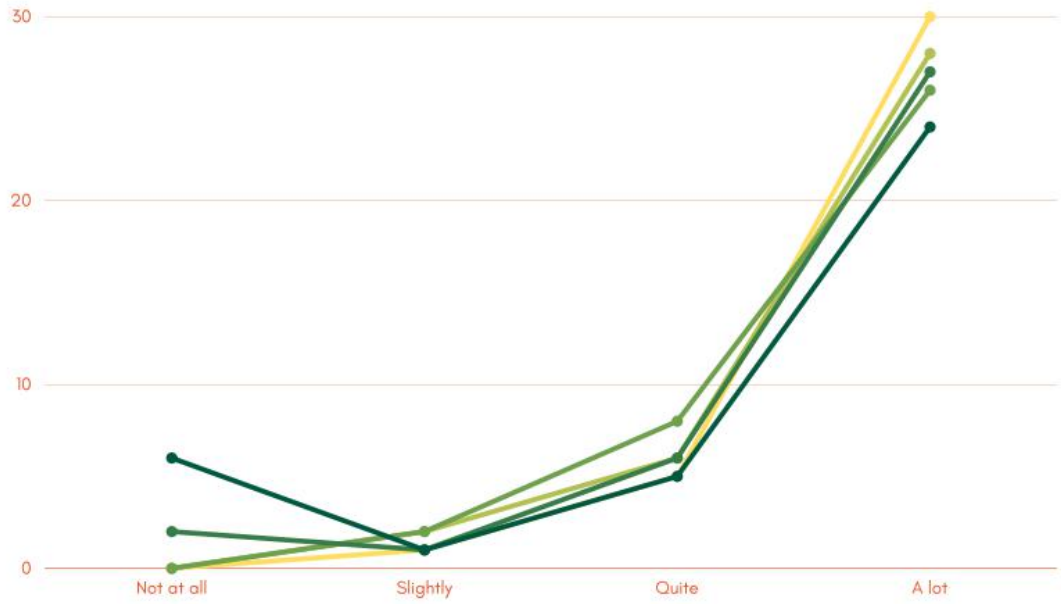
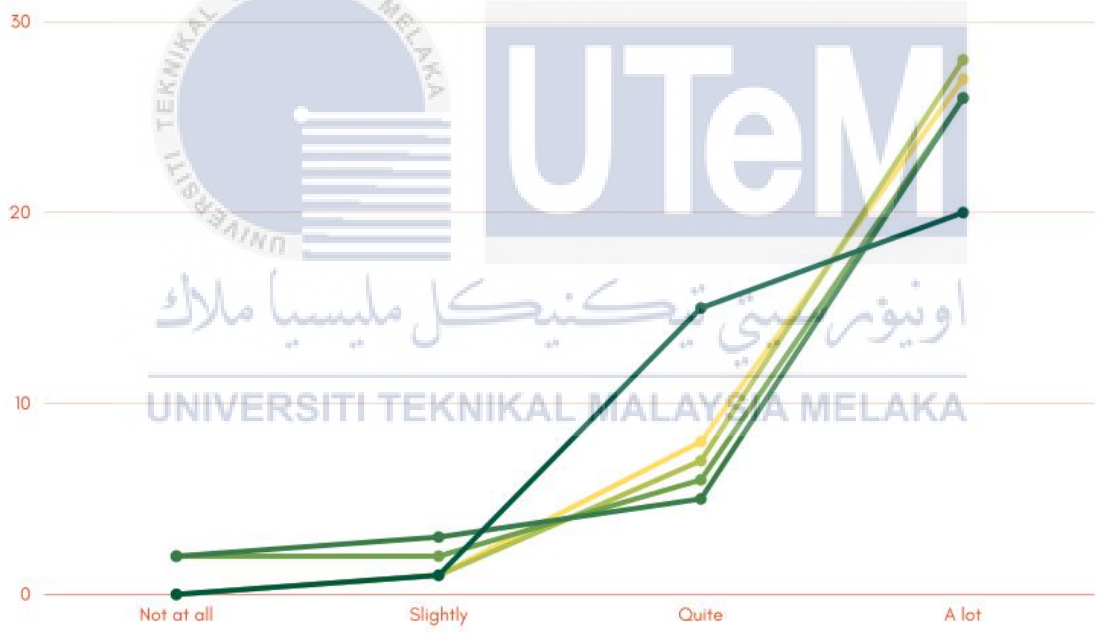


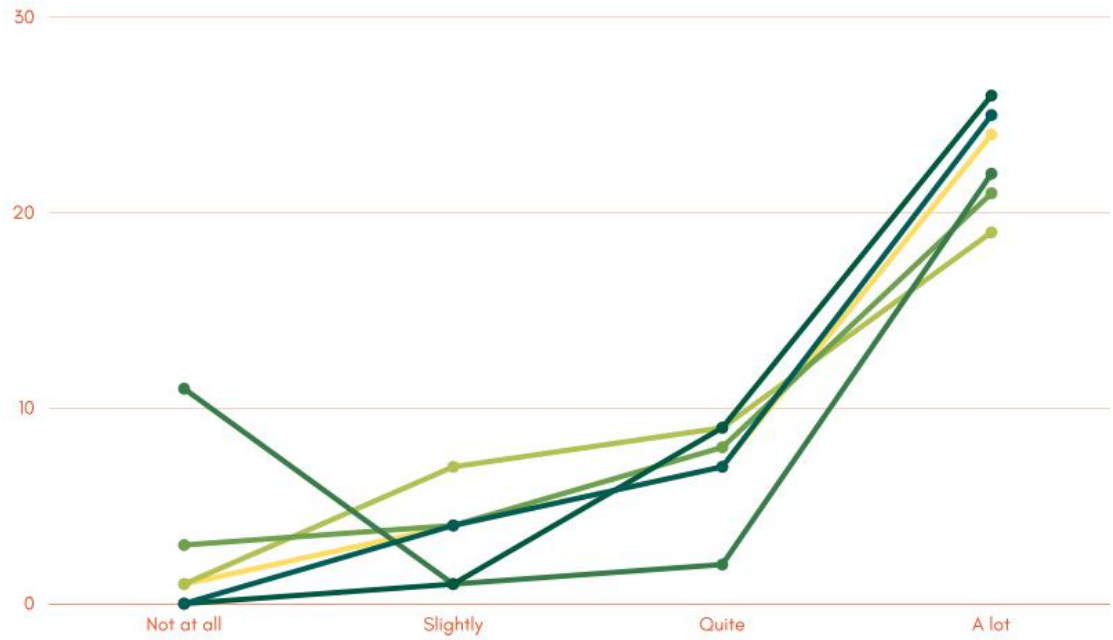
Figure 6.20 Reasons data 2



Data analysis 1 until 5 from reason data



Data analysis 6-10 from reason data



Data analysis 11-16 from reason data

Figure 6.19 and Figure 6.20 shows the reasons that make the respondents keep playing a game again. The highest of "a lot" is the respondents want to see how game ends, re-experience emotions the game made you feel and to master the game. The highest reasons of "quite" is to satisfy nostalgia, to play with friends and to find bugs or ways to break the game. The highest of "not at all" is to explore DLC(downloadable content) and to explore different ways to affect a story endings.

6.5. Conclusion

The conclusion is the chapter explain about the how the testing is done and results from the questionnaire after the testing the game.

As the conclusion, this chapter briefly explain about the testing and results from the respondents after engaging through the game itself or by watching the walktrough video before answering questionnaires. There is a total of 19 questions and 30 respondents participated. The results shown that majority of the respondents strongly agree that the game are positive in every aspects. Based on the respondents recommendation, the game can be improve through presentation and aesthetics aspects such as realistic graphics and audio element in the game that involve ambient sound and voiceover. In the next chapter, the observation of strength and weaknesses, proposition for improvement and contribution will be discussed.

CHAPTER 7

PROJECT CONCLUSION

7. 1. Introduction

In this final chapter, discussion will be made to summarize this project. The important points discussed including project summarization, project contribution, project limitation, future works and conclusion.

7. 2. Observation of Strength and Weakness

The character progression mechanics in my projects are one of their strongest features, as they allow players to improve their characters' abilities. For example, players can buy upgrades from the shop to improve any abilities they want. This mechanic is important because it emphasis the game's replay value, as the player can play the game again to improve their skills and fight more difficult enemies.

The project's flaw is that the gameplay is too simple, and as a result, the player may become bored quickly. The game lacks complexity in terms of what the player can do, and after a few plays, the game becomes quite predictable.

7. 3. Proposition for Improvement

Based on player feedback, the game requires more things that players can discover later on. Adding more items for the player to use will increase the game's replayability. Having a different boss or a special boss can increase the player's excitement by making the level design more complex and changing the layout of the levels every time they play the game.

7.4. Contribution

This project contributes to further the studies on integrating game replay values in video games. This contribute to the innovation on how video games is developed to have game replay value attraction. This project also serve as a model for any future video game development so that developers can evaluate the video game potential and quality to be played after the first completion.

7.5. Conclusion

As a result, this project met its goals of researching, developing, and evaluating how game replay values are integrated into video games so that the game has the potential or value to be played again after the first completion. This project also provides benefits to game developers for any future game development.

REFERENCES

Analysis: The Roguelike Chronicles - Chocobo's Dungeon for Wii. (n.d.). Retrieved from https://www.gamecareerguide.com/news/119680/analysis_the_roguelike_chronicles_.php

BEEP BOOP ROBOTS DETECTED. (n.d.). Retrieved from https://www.greenmangaming.com/blog/what-is-a-roguelike/

Bycer, J. (2021, August 25). The Roguelike Debate -- Roguelikes vs Roguelites. Retrieved from https://www.gamasutra.com/blogs/JoshBycer/20191125/354673/The_Roguelike_Debate__Roguelikes_vs_Roguelites.php

Cataclysm Dark Days Ahead: Static Analysis and Roguelike Games. (n.d.). Retrieved from https://medium.com/pvs-studio/cataclysm-dark-days-ahead-static-analysis-and-roguelike-games-7c9d885b0741

Enter the Gungeon Wiki. (n.d.). Retrieved from https://enterthegungeon.fandom.com/wiki/Enter_the_Gungeon_Wiki

John HarrisContributorFebruary 02, 2. (n.d.). Analysis: The Eight Rules Of Roguelike Design. Retrieved from https://www.gamasutra.com/view/news/123031/Analysis_The_Eight_Rules_Of_Roguelike_Design.php

King, A. (2021, March 18). The Key Design Elements of Roguelikes. Retrieved from https://gamedevelopment.tutsplus.com/articles/the-key-design-elements-of-roguelikes--cms-23510

Kudgel. (2007, October 30). Replay value : How to add to the experience? Retrieved from https://gamedev.net/forums/topic/470300-replay-value-how-to-add-to-the-experience/470300/

Like, T. R., & Bycer, J. (2019, November 22). Breaking the RogueLike Wall - Roguelikes vs Rogue-lites. Retrieved from https://game-wisdom.com/critical/roguelikes-vs-rogue-lites

On Mars: An Analysis of Replay Value: On Mars. (n.d.). Retrieved from https://boardgamegeek.com/thread/2342018/mars-analysis-replay-value

Replayability: Game Mechanics As Periodic Dilemma Generators. (n.d.). Retrieved from https://www.gamecareerguide.com/features/1981/replayability_game_mechanics_as_.php

Slash. (2007, July 02). Failure rates of Roguelike Games. Retrieved from https://blog.roguetemple.com/2007/07/01/failure-rates-of-roguelike-games/

SnapShot. (2020, February 23). About "Replay Value" in Video Games. Retrieved from https://neogaf.com/threads/about-replay-value-in-video-games.1527450/

What Gives a Game "Replay" Value? (n.d.). Retrieved from https://www.stardock.com/blog/505723/what-gives-a-game-replay-value

Roth, Christian & Vermeulen, Ivar & Vorderer, Peter & Klimmt, Christoph. (2012). Exploring Replay Value: Shifts and Continuities in User Experiences Between First and Second Exposure to an Interactive Story. Cyberpsychology, behavior and social networking. 15. 378-81. 10.1089/cyber.2011.0437. Retrieved from https://www.researchgate.net/publication/229063702_Exploring_Replay_Value_Shifts_and_Continuities_in_User_Experiences_Between_First_and_Second_Exposure_to_an_Interactive_Story

APPENDIX
Appendix A :
Questionnaires

Analyzing Game Replay Values in Video Games

Hi ! Thank you for your participation and feedback.

This project focus on developing a game that integrated with game replay values. This game was developed to keep the player entertained and keep coming back to play the game after the first completion. The genre is rogue-like which is a sub-genre to role playing game(RPG) genre. This project is developed to design systems or mechanics to keep player playing the game without feel bored. Thus, the outcome for this project is to design systems and mechanics to integrate it with the game to enhance the game replay values.

*Required

1. Gender *

Mark only one oval.

- Male
- Female

2. Age *

Mark only one oval.

- 18-20 years old
- 21-24 years old
- 25-30 years old
- 30 years old and above

3. Do you play a lot of video game? If so how do you rate yourself? *

Mark only one oval.

- ◯ Casual
- ◯ Hardcore
- ◯ Not a gamer

Rate Your Experience

Rate your experience playing the game Dungeon Revive

4. I find it is easy to learn how to play this game *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

5. I find the instruction provided in the game is clear *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

6. I find the game's menus are user friendly *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

7. I find the game is easy to play *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

8. I find the game is very challenging *

Mark only one oval.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

9. What did you like about the game? *

Tick all that apply.

- [ ] gameplay mechanics
- [ ] level design
- [ ] art style
- [ ] currency system
- [ ] level progression
- [ ] character progression
- [ ] sound and music

Other: _____

10. How much did you like the character progression system? *

Mark only one oval.

	1	2	3	4	5	
Extremely Low	◯	◯	◯	◯	◯	Extremely High

11. How much did you like the level progression system? *

Mark only one oval.

	1	2	3	4	5	
Extremely Low	◯	◯	◯	◯	◯	Extremely High

12. How much did you like the shop/currency system? *

Mark only one oval.

	1	2	3	4	5	
Extremely Low	◯	◯	◯	◯	◯	Extremely High

13. If you have beaten the last boss, did you feel like wanted to play again? *

14. Tell your experience after second time playing the game? (Does the game feel different or feel the same) *

15. have you play any rogue-like game genre? *

Tick all that apply.

☐ hades ☐ enter the gungeon

☐ dead cells ☐ the binding of isaac

Other: ☐ _____

☐ No

16. What other games did you play? *

_____

17. What makes you "addicted" to the game? *

Tick all that apply.

- [ ] gameplay mechanics
- [ ] level design
- [ ] art style
- [ ] currency system
- [ ] level progression
- [ ] character progression
- [ ] story progression
- [ ] sound and music
- [ ] competitive gameplay
- [ ] Online play
- [ ] Other: _____

18. If you encounter a part of a game that is very difficult for you, which best describes how you behave? *

Tick all that apply.

- [ ] I look for a different game to play
- [ ] I lower the difficulty of the game
- [ ] Whether I keep trying depends on how much I have spent on the game (time, money and effort)
- [ ] I keep trying until I feel like I'm not making enough progress
- [ ] I look for walkthroughs or cheats
- [ ] Other: _____

19. From 1 to 4, how much does each of the following reasons make you feel like playing a game again? * *

Mark only one oval per row.

	1 (Not at all)	2 (Slightly)	3 (Quite)	4 (A lot)
To see how the game ends				
To re-experience emotions the game made you feel				
To see something you like again (Could be anything)				
To explore everything you can do in the game				
To explore different ways you can affect the story's ending or outcome				
To challenge yourself				
To master the game				
To explore different ways you can achieve the same outcome (play strategies)				
To complete all the official achievements				
To satisfy nostalgia				
To do something creative in the game				
To find bugs and ways to "break" the game				
To try out different game modes				
To explore new DLC				
To play with your friends again				
To interact again with new strangers online				

Suggestion for improvement:

32 responses

Great Work!

more characters

more levels

The gameplay stil need to be fixed

Make level design more complex , player can teleport back to base at any level

More upgrade variations

everything is perfect

Great, Good, Wonderful

Adding more item for the player can increase the replayability of the game. Having different boss or special boss also can make the excitement last longer for player.

Suggestion for improvement:

32 responses

make more things to explore

The controls is ok, but need to be improve

The player control feels clunky

This is game is for hardcore but not made for casual players

The mechanic can still be better

The player control is weird and the UI need to be where player can see easily

The UI need to be fixed

UI position need to be fixed

make the combat more smooth

Suggestion for improvement:

32 responses

more fun

great!

The best

need more levels

Nice

nice

nice work!

More Levels

not enough things to discover

Appendix B: Coding Script

Game controller script

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

public class GameController : MonoBehaviour
{
    public GameObject player;
    public GameObject playerDieScreen;
    public GameObject playerHurtFX;
    public GameObject pauseMenuUI, SettingUI,storyUI;

    private static float health = 50;
    private static int maxHealth = 50;
    private static int coin = 100;
    private static int enemyKilledCount = 0;
    private static float moveSpeed = 2f;
    private static float fireRate = 0.5f;
    private static float bulletSize = 1f;

    int escapedCount =  0;
    bool playerescaped =false;

    public static bool isHurted = false;

    public bool isDead = false;
    public bool isPaused = false;
    public bool isHealthMaxxed = false;


    public static float Health { get => health; set => health = value; }
    public static int MaxHealth { get => maxHealth; set => maxHealth = value; }
    public static int Coin { get => coin; set => coin = value; }
    public static float MoveSpeed { get => moveSpeed; set => moveSpeed = value; }
    public static float FireRate { get => fireRate; set => fireRate = value; }
    public static int EnemyKilledCount { get => enemyKilledCount; set => enemyKilledCount = value; }
    public static float BulletSize { get => bulletSize; set => bulletSize = value; }


    public Text healthText;
    public Text coinText;
    public Text speedText;
    public Text enemykillText;
    public static bool isPlayerDead=false;
    //public GameObject deathPanel;


    // Start is called before the first frame update
    void Start()
    {
        pauseMenuUI.SetActive(false);
        playerHurtFX.SetActive(false);
        playerDieScreen.SetActive(false);
    }

    // Update is called once per frame
```

```
void Update()
{
    healthText.text = "Health :" + health;
    coinText.text = "Coins : " + coin;
    speedText.text = "Speed : " + moveSpeed;
    enemykillText.text = "Enemies killed :" + enemyKilledCount;

    if (isPlayerDead==true)
    {
        //deathPanel.SetActive(true);
        Time.timeScale = 0f;
    }
    else
    {
        //deathPanel.SetActive(false);
        Time.timeScale = 1f;
    }

    if (Health <= 0)
    {
        isDead = true;
        PlayerDieScreenSpawn();
        player.SetActive(false);
    }

     if (isPlayerDead == true)
    {
        if (Input.GetKey(KeyCode.R))
        {
            SceneManager.LoadScene("Lobby");
            //SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
            isPlayerDead = false;
            health = maxHealth;
            StartCoroutine(afterspawn());
            BossHealthController.playerRespawned(true);
            BossHealthController2.playerRespawned(true);
            BossHealthController3.playerRespawned(true);
            BossHealthController4.playerRespawned(true);
        }
    }



    if (Input.GetKey(KeyCode.M))
    {
        SceneManager.LoadScene("MENU");
    }

    //if (Input.GetKey(KeyCode.Escape))
    //{
    //    Application.Quit();
    //}



    if (isHurted == true)
    {
        StartCoroutine(playerHurted());
    }

    if (Input.GetKey(KeyCode.Escape))
    {
        pauseMenuUI.SetActive(true);
```

```
    isPaused = true;

}

if (isPaused==true)
{
    Time.timeScale = 0f;
}

if (escapedCount==1)
{
    enemyController.speedUpgrade(true);
    enemyController.hitCountUp(1);
    bossController.canSpawnenemy(true);
    bossController2.canSpawnenemy(true);
    bosscontroller3.canSpawnenemy(true);
    bosscontroller4.canSpawnenemy(true);
}


if (escapedCount == 2)
{
    enemyController.speedUpgrade(true);
    enemyController.hitCountUp(1);
    bossController.canSpawnenemy(true);
    bossController2.canSpawnenemy(true);
    bosscontroller3.canSpawnenemy(true);
    bosscontroller4.canSpawnenemy(true);
}

//--------cheatcode----------
if (Input.GetKeyDown(KeyCode.L))
{
    coin += 100;
}

if (Input.GetKeyDown(KeyCode.O))
{
    enemyKilledCount += 10;
}


if (Input.GetKeyDown(KeyCode.Alpha1))
{
    SceneManager.LoadScene("level1_bossRoom");
}
if (Input.GetKeyDown(KeyCode.Alpha2))
{
    SceneManager.LoadScene("level2_bossRoom");
}
if (Input.GetKeyDown(KeyCode.Alpha3))
{
    SceneManager.LoadScene("level3_bossRoom");
}
if (Input.GetKeyDown(KeyCode.Alpha4))
{
    SceneManager.LoadScene("lastroom");
}
if (Input.GetKeyDown(KeyCode.Alpha5))
{
    Escaped();
}
//-----------------------------
```

```
}

public static void DamagePlayer(int damage)
{
   health -= damage;

   if (Health <= 0)
   {
      KillPlayer();

      isPlayerDead = true;
   }
}

public static void KillPlayer()
{

}

IEnumerator afterspawn()
{
   yield return new WaitForSeconds(2f);
   Time.timeScale = 1f;
   isPlayerDead = false;
}

public static void HealPlayer(float healAmount)
{
   health = Mathf.Min(maxHealth, health + healAmount);
}

public static void MoveSpeedChange(float speed)
{
   moveSpeed += speed;
}

public static void FireRateChange(float rate)
{
   fireRate -= rate;
}

public static void BulletSizeChange(float size)
{
   bulletSize += size;
}



public static void currencyAdd(int money)
{
   coin += money;

}

public static void currencyMinus(int money)
{
   coin -= money;
   // coinCost.text = "Coins : " + coin;
}

public static void addenemykillCount(int EnemyNum)
{
   enemyKilledCount += EnemyNum;
```

```
}

IEnumerator playerHurted()
{
   playerHurtFX.SetActive(true);
   yield return new WaitForSeconds(0.5f);
   playerHurtFX.SetActive(false);
   isHurted = false;
}


void PlayerDieScreenSpawn()
{
   playerDieScreen.SetActive(true);
   //Time.timeScale = 0f;
   isPaused = true;
}

public static void plusMaxHealth(int maxHealthPlus)
{
   maxHealth += maxHealthPlus;
   health += maxHealthPlus;

}

public static void plusmoveSpeed(float MoveSpeed)
{
   moveSpeed += MoveSpeed;
}



public void Setting()
{
   SettingUI.SetActive(true);
}
public void BACKSetting()
{
   SettingUI.SetActive(false);
}

public void offstory()
{
   storyUI.SetActive(false);
}

public void Resume()
{
   pauseMenuUI.SetActive(false);
   isPaused = false;
}
public void LoadMenu()
{
   //Time.timeScale = 1f;
   pauseMenuUI.SetActive(false);
   SceneManager.LoadScene("MENU");

}

public void Escaped()
{
   //Time.timeScale = 1f;
```

```
        pauseMenuUI.SetActive(false);
        SceneManager.LoadScene("MENU");
        playerescaped = true;
        escapedCount += 1;
        BossHealthController.playerRespawned(true);
        BossHealthController2.playerRespawned(true);
        BossHealthController3.playerRespawned(true);
        BossHealthController4.playerRespawned(true);


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


}
```

Player Controller

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

public class playerController : MonoBehaviour
{
    public float speed;
    //public float dashSpeed;
    Rigidbody2D rb;
    public Text collectedText;
    public static int collectedAmount = 0;
    public GameObject firepointer;

    public Animator animator;



    bool canDash = false;
    public int dashCD = 80;

    public Camera cam;
    Vector2 movement;
    Vector2 mousePos;
    Vector2 posDif;
    Vector2 moveDir;

    private static playerController instance;

    public static playerController PlayerInstance
    {
        get
        {
            if (instance == null)
            {
                instance = FindObjectOfType<playerController>();
            }
            return instance;
        }
    }



    // Start is called before the first frame update
    void Start()
    {

        rb = GetComponent<Rigidbody2D>();
    }

    void Update()
    {
        speed = GameController.MoveSpeed;
        movement.x = Input.GetAxis("Horizontal");
        movement.y = Input.GetAxis("Vertical");
        moveDir = new Vector2(movement.x, movement.y).normalized;
        mousePos = cam.ScreenToWorldPoint(Input.mousePosition);

        posDif = mousePos - rb.position;
        animator.SetFloat("Mouse Horizontal", posDif.x);
        animator.SetFloat("Mouse Vertical", posDif.y);
```

```
        animator.SetFloat("walk Hor", movement.x);
        animator.SetFloat("walk Ver", movement.y);
        animator.SetFloat("moving", moveDir.magnitude);


        //animator.SetBool("isWalk", true);

        if (Input.GetKey(KeyCode.Space) && canDash == true)
        {
            rb.MovePosition(transform.position + (transform.forward * (speed *
Time.deltaTime)));
            rb.AddForce(movement.normalized * 6000);
            canDash = false;
            dashCD = 80;

        }



    }

    void FixedUpdate()
    {
        rb.MovePosition(rb.position + movement * speed * Time.fixedDeltaTime);
        Vector2 lookDir = mousePos - rb.position;
        float angle = Mathf.Atan2(lookDir.y, lookDir.x) * Mathf.Rad2Deg - 90f;
        //rb.rotation = angle;

        if (dashCD == 0)
        {
            canDash = true;
        }
        else
        {
            dashCD--;
        }
    }

    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "Enemy")
        {
            GameController.DamagePlayer(10);
        }
    }
}
```

Shop script

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

public class shopPanel : MonoBehaviour
{

   public GameObject panel;
   public Text coinCost;
   private static int coinHealthUp =100;
   public int healthUpCounter=0;
   public Text coinCostSpeed;
   private static int coinSpeedUp = 100;
   public int speedUpCounter = 0;
   public Text coinCosttravel;
   private static int cointravelUp = 100;
   public int travelUpCounter = 0;

   public Text upgrade1, upgrade2, upgrade3;

   // Start is called before the first frame update
   void Start()
   {
      panel.SetActive(false);
   }

   // Update is called once per frame
   void Update()
   {
      coinCost.text = "COST :" + coinHealthUp;
      coinCostSpeed.text = "COST :" + coinSpeedUp;
      coinCosttravel.text = "COST :" + cointravelUp;

   }

    void OnTriggerEnter2D(Collider2D collision)
   {

      if (collision.tag=="Player")
      {
         panel.SetActive(true);
      }

   }

    void OnTriggerExit2D(Collider2D collision)
   {

      if (collision.tag == "Player")
      {
         panel.SetActive(false);
      }
   }

   //health upgrades-----------------------------------------------------------------
   public void HealthUpgrade()
   {

      if (healthUpCounter==0)
      {
         if (GameController.Coin >= 100)
         {
```

```
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusMaxHealth(10);
            coinHealthUp += 100;
            healthUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {


            Debug.Log("no money");
            GameController.plusMaxHealth(0);
        }
    }
    if (healthUpCounter == 1)
    {
        if (GameController.Coin >= 200)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusMaxHealth(10);
            coinHealthUp += 100;
            healthUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            GameController.plusMaxHealth(0);
        }
    }
    if (healthUpCounter == 2)
    {
        if (GameController.Coin >= 300)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusMaxHealth(10);
            coinHealthUp += 100;
            healthUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            GameController.plusMaxHealth(0);
        }
    }
    if (healthUpCounter == 3)
    {
        if (GameController.Coin >= 400)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusMaxHealth(10);
            coinHealthUp += 100;
            healthUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            GameController.plusMaxHealth(0);
        }
    }
    if (healthUpCounter == 4)
    {
        if (GameController.Coin >= 500)
        {
            SoundScript.Playsound("upgrade");
```

```
            GameController.currencyMinus(100);
            GameController.plusMaxHealth(10);
            coinHealthUp += 100;
            healthUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            GameController.plusMaxHealth(0);
        }
    }
    if (healthUpCounter == 5)
    {
        upgrade1.text = " MAX ";
        Debug.Log("Health upgrade is max");
        SoundScript.Playsound("nomoney");
    }
}


//SPEED UPGRADES---------------------------------------
public void speedUpgrade()
{
    if (speedUpCounter == 0)
    {
        if (GameController.Coin >= 100)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusmoveSpeed(1);
            coinSpeedUp += 100;
            speedUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            GameController.plusmoveSpeed(0);
        }
    }
    if (speedUpCounter == 1)
    {
        if (GameController.Coin >= 200)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusmoveSpeed(1);
            coinSpeedUp += 100;
            speedUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            GameController.plusmoveSpeed(0);
        }
    }
    if (speedUpCounter == 2)
    {
        if (GameController.Coin >= 300)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusmoveSpeed(1);
            coinSpeedUp += 100;
            speedUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
```

```
            Debug.Log("no money");
            GameController.plusmoveSpeed(0);
        }
    }
    if (speedUpCounter == 3)
    {
        if (GameController.Coin >= 400)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusmoveSpeed(1);
            coinSpeedUp += 100;
            speedUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            GameController.plusmoveSpeed(0);
        }
    }
    if (speedUpCounter == 4)
    {
        if (GameController.Coin >= 500)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            GameController.plusmoveSpeed(1);
            coinSpeedUp += 100;
            speedUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            GameController.plusmoveSpeed(0);
        }
    }
    if (speedUpCounter == 5)
    {
        upgrade2.text = " MAX ";
        Debug.Log("Speed upgrade is max");
        SoundScript.Playsound("nomoney");
    }
}


//===============Bullet upgrades ==========================Bullet upgrades ==================================Bullet upgrades ====
public void BulletTravelUp()
{
    if (travelUpCounter == 0)
    {
        if (GameController.Coin >= 100)
        {
            SoundScript.Playsound("upgrade");
            GameController.currencyMinus(100);
            bulletController.lifetimechange(0.1f);
            cointravelUp += 100;
            travelUpCounter += 1;
        }
        else if (GameController.Coin == 0)
        {
            Debug.Log("no money");
            bulletController.lifetimechange(0);
        }
    }
    if (travelUpCounter == 1)
    {
```

```
    if (GameController.Coin >= 200)
    {
        SoundScript.Playsound("upgrade");
        GameController.currencyMinus(100);
        bulletController.lifetimechange(0.1f);
        cointravelUp += 100;
        travelUpCounter += 1;
    }
    else if (GameController.Coin == 0)
    {
        Debug.Log("no money");
        bulletController.lifetimechange(0);
    }
}
if (travelUpCounter == 2)
{
    if (GameController.Coin >= 300)
    {
        SoundScript.Playsound("upgrade");
        GameController.currencyMinus(100);
        bulletController.lifetimechange(0.1f);
        cointravelUp += 100;
        travelUpCounter += 1;
    }
    else if (GameController.Coin == 0)
    {
        Debug.Log("no money");
        bulletController.lifetimechange(0);
    }
}
if (travelUpCounter == 3)
{
    if (GameController.Coin >= 400)
    {
        SoundScript.Playsound("upgrade");
        GameController.currencyMinus(100);
        bulletController.lifetimechange(0.1f);
        cointravelUp += 100;
        travelUpCounter += 1;
    }
    else if (GameController.Coin == 0)
    {
        Debug.Log("no money");
        bulletController.lifetimechange(0);
    }
}
if (travelUpCounter == 4)
{
    if (GameController.Coin >= 500)
    {
        SoundScript.Playsound("upgrade");
        GameController.currencyMinus(100);
        bulletController.lifetimechange(0.1f);
        cointravelUp += 100;
        travelUpCounter += 1;
    }
    else if (GameController.Coin == 0)
    {
        Debug.Log("no money");
        bulletController.lifetimechange(0);
    }
}
if (travelUpCounter == 5)
{
    upgrade3.text = " MAX ";
    Debug.Log("travel time upgrade is max");
    SoundScript.Playsound("nomoney");
```

```
    }


  }

  public void UnlimitedMagic()
  {

    if (GameController.Coin >= 1000)
    {
      SoundScript.Playsound("upgrade");
      GameController.currencyMinus(1000);
      playershooting.setUnlimited(true);
    }
  }}
```

Achievement panel script

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

public class achievementPanel : MonoBehaviour
{
    public GameObject AchievementPanelMenu;

    public GameObject complete1, notcomplete1;
    public GameObject complete2, notcomplete2;
    public GameObject complete3, notcomplete3;
    public GameObject complete4, notcomplete4;
    public GameObject complete5, notcomplete5;

    bool b1disable = false;
    bool b2disable = false;
    bool b3disable = false;
    bool b4disable = false;
    bool b5disable = false;

    void Start()
    {
        AchievementPanelMenu.SetActive(false);

        notcomplete1.SetActive(true);
        complete1.SetActive(false);

        notcomplete2.SetActive(true);
        complete2.SetActive(false);

        notcomplete3.SetActive(true);
        complete3.SetActive(false);

        notcomplete4.SetActive(true);
        complete4.SetActive(false);

        notcomplete5.SetActive(true);
        complete5.SetActive(false);


    }


    void Update()
    {
```

```
}


void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.tag == "Player")
    {
        AchievementPanelMenu.SetActive(true);
    }
}

void OnTriggerExit2D(Collider2D collision)
{
    if (collision.tag == "Player")
    {
        AchievementPanelMenu.SetActive(false);
    }
}

public void achieve1Complete()
{
    if (GameController.EnemyKilledCount >= 20 && b1disable==false)
    {
        GameController.currencyAdd(200);
        notcomplete1.SetActive(false);
        complete1.SetActive(true);
        b1disable = true;
    }}
public void achieve2Complete()
{
    if (GameController.EnemyKilledCount >= 40 && b2disable == false)
    {
        GameController.currencyAdd(300);
        notcomplete2.SetActive(false);
        complete2.SetActive(true);
        b2disable = true;
  }}
public void achieve3Complete()
{
    if (GameController.EnemyKilledCount >= 60 && b3disable == false)
    {
        GameController.currencyAdd(400);
        notcomplete3.SetActive(false);
        complete3.SetActive(true);
        b3disable = true;
    }}
public void achieve4Complete()
{
    if (GameController.EnemyKilledCount >= 80 && b4disable == false)
    {
        GameController.currencyAdd(500);
        notcomplete4.SetActive(false);
        complete4.SetActive(true);
        b4disable = true;


    }
}

public void achieve5Complete()
{
    if (GameController.EnemyKilledCount >= 100 && b5disable == false)
    {
        GameController.currencyAdd(600);
        notcomplete5.SetActive(false);
```

```
            complete5.SetActive(true);
            b5disable = true;


        }
    }
}
```

Boss controller script

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

public class bossController : MonoBehaviour
{
    public bool bossEnemy;

    public GameObject bosshealthBar, entergate,mobspawner;
    public Transform firePoint, firePoint2, firePoint3, firePoint4;
    public Transform firePoint5, firePoint6, firePoint7, firePoint8;
    public GameObject bulletPrefab;

    public GameObject HealPrefab;
    public GameObject coinUIprefab;
    public GameObject slimeexplodeprefab;

    public float bulletForce = 15f;
    public float cd=2f;
    private bool cdAttack = false;
    public float cd2 = 1f;
    private bool cdAttack2 = false;
    int rewardCounter = 0;

    private static bool isattacked = false;
    private static bool isLow = false;
    private static bool isBossdead = false;

    public static bool ISattacked { get => isattacked; set => isattacked = value; }
    public static bool ISLow { get => isLow ; set => isLow = value; }
    public static bool ISBossdead { get => isBossdead; set => isBossdead = value; }
    private static bool canSpawnMob = false;
    public static bool CanSpawnMob { get => canSpawnMob; set => canSpawnMob =
value; }

    void Start()
    {
        //bosshealthBar.SetActive(false);
        mobspawner.SetActive(false);
    }


    private void OnTriggerEnter2D(Collider2D col)
    {
        if (col.tag == "playerBullet")
        {
            BossHealthController.BossDamaged(5);
            //bosshealthBar.SetActive(true);
        }
    }
```

```
    void Update()
    {
        if (bossEnemy == true)
        {
            enemybulletController.lifetimechange(3f);
        }


        if (ISattacked==true)
        {
            Attack();
            entergate.SetActive(false);
        }

        if (ISLow==true)
        {
            Attack2();
            if (canSpawnMob==true)
            {
                mobspawner.SetActive(true);
            }

        }

        if (isBossdead==true)
        {

            rewardCounter += 1;

        }
        else
        {
            rewardCounter =0;
        }


        if (rewardCounter==1)
        {
            //int coindrop = Random.Range(1, 100);
            //GameController.currencyAdd(coindrop);
            //GameObject coinprefab = Instantiate(coinUIprefab, transform.position,
Quaternion.identity) as GameObject;
            //GameObject     slimeexplode     =     Instantiate(slimeexplodeprefab,
transform.position, Quaternion.identity) as GameObject;
            //coinUIplus.addcoinvalue(coindrop);
            //Destroy(slimeexplode, 0.2f);
            //Destroy(coinprefab, 1f);
            //GameObject   heal   =   Instantiate(HealPrefab,   transform.position,
Quaternion.identity) as GameObject;
        }


    }

    public static void IsbossAttacked(bool attacked)
    {
        ISattacked = attacked;
    }

    public static void IsHealthLow(bool healthcondition)
    {
        ISLow = healthcondition;
    }

    public static void IsbossDead(bool bossDead)
```

```
    {
        isBossdead = bossDead;
    }

    public void Attack()
    {
        if (!cdAttack)
        {
            SoundScript.Playsound("enemyfire");
            GameObject bullet = Instantiate(bulletPrefab, firePoint.position,
firePoint.rotation);
            Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();
            bullet.GetComponent<enemybulletController>().isEnemyBullet = true;
            rb.AddForce(firePoint.up * bulletForce, ForceMode2D.Impulse);
            Destroy(bullet, 2f);

            GameObject bullet2 = Instantiate(bulletPrefab, firePoint2.position,
firePoint2.rotation);
            Rigidbody2D rb2 = bullet2.GetComponent<Rigidbody2D>();
            bullet2.GetComponent<enemybulletController>().isEnemyBullet = true;
            rb2.AddForce(firePoint2.up * bulletForce, ForceMode2D.Impulse);
            Destroy(bullet2, 2f);

            GameObject bullet3 = Instantiate(bulletPrefab, firePoint3.position,
firePoint3.rotation);
            Rigidbody2D rb3 = bullet3.GetComponent<Rigidbody2D>();
            bullet3.GetComponent<enemybulletController>().isEnemyBullet = true;
            rb3.AddForce(firePoint3.up * bulletForce, ForceMode2D.Impulse);
            Destroy(bullet3, 2f);

            GameObject bullet4 = Instantiate(bulletPrefab, firePoint4.position,
firePoint4.rotation);
            Rigidbody2D rb4 = bullet4.GetComponent<Rigidbody2D>();
            bullet4.GetComponent<enemybulletController>().isEnemyBullet = true;
            rb4.AddForce(firePoint4.up * bulletForce, ForceMode2D.Impulse);
            Destroy(bullet4, 2f);
            StartCoroutine(Cooldown());
        }
    }

    public void Attack2()
    {
        if (!cdAttack2)
        {
            GameObject bullet5 = Instantiate(bulletPrefab, firePoint5.position,
firePoint5.rotation);
            Rigidbody2D rb5 = bullet5.GetComponent<Rigidbody2D>();
            bullet5.GetComponent<enemybulletController>().isEnemyBullet = true;
            rb5.AddForce(firePoint5.up * bulletForce, ForceMode2D.Impulse);
            Destroy(bullet5, 2f);

            GameObject bullet6 = Instantiate(bulletPrefab, firePoint6.position,
firePoint6.rotation);
            Rigidbody2D rb6 = bullet6.GetComponent<Rigidbody2D>();
            bullet6.GetComponent<enemybulletController>().isEnemyBullet = true;
            rb6.AddForce(firePoint6.up * bulletForce, ForceMode2D.Impulse);
            Destroy(bullet6, 2f);

            GameObject bullet7 = Instantiate(bulletPrefab, firePoint7.position,
firePoint7.rotation);
            Rigidbody2D rb7 = bullet7.GetComponent<Rigidbody2D>();
            bullet7.GetComponent<enemybulletController>().isEnemyBullet = true;
            rb7.AddForce(firePoint7.up * bulletForce, ForceMode2D.Impulse);
            Destroy(bullet7, 2f);
```

```
            GameObject bullet8 = Instantiate(bulletPrefab, firePoint8.position,
firePoint8.rotation);
            Rigidbody2D rb8 = bullet8.GetComponent<Rigidbody2D>();
            bullet8.GetComponent<enemybulletController>().isEnemyBullet = true;
            rb8.AddForce(firePoint8.up * bulletForce, ForceMode2D.Impulse);
            Destroy(bullet8, 2f);
            StartCoroutine(Cooldown2());
        }
    }



    private IEnumerator Cooldown()
    {
        cdAttack = true;
        yield return new WaitForSeconds(cd);
        cdAttack = false;

    }
    private IEnumerator Cooldown2()
    {
        cdAttack2 = true;
        yield return new WaitForSeconds(cd2);
        cdAttack2 = false;

    }

    private IEnumerator reward()
    {
        yield return new WaitForSeconds(1f);
    }



    public static void canSpawnenemy(bool value)
    {
        CanSpawnMob = value;
    }
}
```

Boss health controller

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

public class BossHealthController : MonoBehaviour
{
    public GameObject stair;
    public GameObject boss;

    int spawnCounter = 0;

    public GameObject bosshealthBar;
    private float fillValue;

    static bool isTriggered = false;

    bool startAttack=false;
    bool isDead = false;

    private static float enemyBossHealth = 500;
    private static int enemyBossMaxHealth = 500;
    private static bool isPlayerRespawned = false;

    public static float EnemyBossHealth { get => enemyBossHealth; set =>
enemyBossHealth = value; }
    public static int EnemyBossMaxHealth { get => enemyBossMaxHealth; set =>
enemyBossMaxHealth = value; }
    public static bool ISPlayerRespawned { get => isPlayerRespawned; set =>
isPlayerRespawned = value; }

    void Start()
    {
        stair.SetActive(false);
    }

    void Update()
    {
        fillValue = enemyBossHealth;
        fillValue = fillValue / enemyBossMaxHealth;
        bosshealthBar.GetComponent<Image>().fillAmount = fillValue;


        if (Input.GetKeyDown(KeyCode.P))
        {
            enemyBossHealth -= 50;

        }
        if (Input.GetKeyDown(KeyCode.K))
        {

            stair.SetActive(true);
        }

        if (enemyBossHealth <= 0)
        {
            isDead = true;
            boss.SetActive(false);
            bossController.IsbossDead(true);
            //stair.SetActive(true);
```

```
    }

    if (enemyBossHealth >=0)
    {
        isDead = false;
    }

    //if (isDead==true)
    //{
    //    stair.SetActive(true);
    //}
    //else
    //{
    //    stair.SetActive(false);
    //}

    if (ISPlayerRespawned == true)
    {
        isDead = false;
        spawnCounter += 1;



    }

    if (spawnCounter==1)
    {

       // stair.SetActive(false);
        boss.SetActive(true);
        isTriggered = false;
        bossController.IsbossDead(false);
        notlowHealth();
        spawnCounter =0;
    }

    if (enemyBossHealth<=250)
    {
        lowHealth();
    }

     if (enemyBossHealth < 500)
    {
        isTriggered = true;
    }


    if (isTriggered==true)
    {
        startAttack = true;
        bossController.IsbossAttacked(true);
    }
    else
    {
        startAttack = false;
        bossController.IsbossAttacked(false);
    }



}

public static void BossDamaged(float damage)
```

```
    {
        enemyBossHealth -= damage;

    }


    public void lowHealth()
    {
        bossController.IsHealthLow(true);
    }
    public void notlowHealth()
    {
        bossController.IsHealthLow(false);
    }


    public static void playerRespawned(bool Playerrespawned)
    {
        ISPlayerRespawned = Playerrespawned;
        EnemyBossHealth = EnemyBossMaxHealth;
    }

    IEnumerator spawned()
    {

        yield return new WaitForSeconds(1f);

    }
}
```

Coin drop script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class coinUIplus : MonoBehaviour
{
    public Text coinplus;

    private static int coindrop;
    public static int Coindrop { get => coindrop; set => coindrop = value; }
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        coinplus.text = "+" + Coindrop;
    }

    public static void addcoinvalue(int coinvalue)
    {
        Coindrop = coinvalue;
    }
}
```

Bullet controller

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

public class bulletController : MonoBehaviour
{
    private static float lifetime = 0.2f;
    public GameObject hitfx,pufffx;
    public bool isEnemyBullet = false;

    public static float Bulletlifetime { get => lifetime; set => lifetime = value; }



    void Start()
    {
        StartCoroutine(DeathDelay());
        if (!isEnemyBullet)
        {
            transform.localScale = new Vector2(GameController.BulletSize,
GameController.BulletSize);
        }

    }

    IEnumerator DeathDelay()
    {
        yield return new WaitForSeconds(lifetime);
        Destroy(gameObject);
        GameObject poofx = Instantiate(pufffx, transform.position,
Quaternion.identity);
```

```
        Destroy(poofx, 0.5f);
    }

    public void OnCollisionEnter2D(Collision2D collision)
    {
        GameObject explodefx = Instantiate(hitfx, transform.position,
Quaternion.identity);
        Destroy(explodefx, 0.1f);
        // Destroy(gameObject);
    }

    void OnTriggerEnter2D(Collider2D col)
    {
        if (col.tag == "Enemy" && !isEnemyBullet)
        {
            col.gameObject.GetComponent<enemyController>().Death();
            Destroy(gameObject);
            GameObject explodefx = Instantiate(hitfx, transform.position,
Quaternion.identity);
            Destroy(explodefx, 0.1f);
            //enemyController.DamageEnemy(5);
        }

        if (col.tag == "Player" && isEnemyBullet)
        {
            GameController.DamagePlayer(10);
            Destroy(gameObject);
        }

        //if (col.tag == "BossEnemy1" && !isEnemyBullet)
        //{
        //    col.gameObject.GetComponent<bossController>().bossDeath();
        //    Destroy(gameObject);
        //        GameObject explodefx = Instantiate(hitfx, transform.position,
Quaternion.identity);
        //    Destroy(explodefx, 0.1f);
        //    //enemyController.DamageEnemy(5);
        //}
    }


    public static void lifetimechange(float Lifetime)
    {
        lifetime += Lifetime;
    }
}
```

Enemy controller script

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

public enum EnemyState
{
    Idle,
    Wander,
    Follow,
    Die,
    Attack
};

public enum EnemyType
{
    Melee,
    Ranged
};

public class enemyController : MonoBehaviour
{
    GameObject player;
    public EnemyState currState = EnemyState.Idle;
    public EnemyType enemyType;
    public float range;
    public float speed;

    public float attackRange;
    public float cd;
    private bool chooseDir = false;
    private bool dead = false;
    private bool cdAttack = false;
    public bool notInRoom = false;
    public bool diamondEnemy = false;
    public bool skullEnemy = false;
    private Vector3 randomDir;
    public GameObject bulletPrefab;
    public GameObject HealPrefab;
    public Transform firePoint;

    public GameObject coinUIprefab;
    public GameObject slimeexplodeprefab;


    public Animator enemyanim;

    private int hitCounter = 0;

    private static int hittoDead = 6;
    public static int HittoDead { get => hittoDead; set => hittoDead = value; }

    private static bool speedupdate = false;
    public static bool speedUPDATE { get => speedupdate; set => speedupdate =
value; }

    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        //coinplusUI.SetActive(false);
```

```
    }

    void Update()
    {
        switch (currState)
        {
            //case (EnemyState.Idle):
            //    Idle();
            //    break;
            case (EnemyState.Wander):
                Wander();
                break;
            case (EnemyState.Follow):
                Follow();
                break;
            case (EnemyState.Die):
                break;
            case (EnemyState.Attack):
                Attack();
                break;


        }

        if (!notInRoom)
        {
            if (isPlayerInRange(range) && currState != EnemyState.Die)
            {
                currState = EnemyState.Follow;
            }
            else if (!isPlayerInRange(range) && currState != EnemyState.Die)
            {
                currState = EnemyState.Idle;
            }

            if (Vector3.Distance(transform.position, player.transform.position) <=
attackRange)
            {
                currState = EnemyState.Attack;

            }
        }
        else
        {
            currState = EnemyState.Idle;
        }


        if (diamondEnemy==true)
        {
            enemybulletController.lifetimechange(2f);
        }

        if (skullEnemy == true)
        {
            enemybulletController.lifetimechange(3f);
        }


        if (speedUPDATE == true)
        {

            //speed += 0.1f;

        }
    }
```

```
    private bool isPlayerInRange(float range)
    {
        return  Vector3.Distance(transform.position,  player.transform.position)  <=
range;
    }

    private IEnumerator ChooseDirection()
    {
        chooseDir = true;
        yield return new WaitForSeconds(Random.Range(2f, 8f));
        randomDir = new Vector3(0, 0, Random.Range(-1, 1));
        Quaternion nextRotation = Quaternion.Euler(randomDir);
        transform.rotation   =   Quaternion.Lerp(transform.rotation,   nextRotation,
Random.Range(0.5f, 2.5f));
        chooseDir = false;
    }
    void Wander()
    {
        if (!chooseDir)
        {
            StartCoroutine(ChooseDirection());
        }

        transform.position += transform.right * speed * Time.deltaTime;
        if (isPlayerInRange(range))
        {
            currState = EnemyState.Follow;
        }
    }
    void Follow()
    {
        transform.position          =          Vector2.MoveTowards(transform.position,
player.transform.position, speed * Time.deltaTime);
    }
    public void Death()
    {
        hitCounter = hitCounter + 1;
        if (hitCounter >= hittoDead)
        {
            dead = true;
            Destroy(gameObject);
            int coindrop = Random.Range(1, 50);
            GameController.currencyAdd(coindrop);
            GameObject  coinprefab  =  Instantiate(  coinUIprefab,  transform.position,
Quaternion.identity) as GameObject;
            GameObject      slimeexplode      =      Instantiate(slimeexplodeprefab,
transform.position, Quaternion.identity) as GameObject;
            coinUIplus.addcoinvalue(coindrop);
            Destroy(slimeexplode, 0.2f);
            Destroy(coinprefab, 1f);

            GameController.addenemykillCount(1);
            int index = Random.Range(1, 5);
            if (index==3)
            {
                GameObject   heal   =   Instantiate(HealPrefab,   transform.position,
Quaternion.identity) as GameObject;
            }
        }
    }
    public void Attack()
    {
        if (!cdAttack)
        {
```

```
            switch (enemyType)
            {
                case (EnemyType.Melee):
                    GameController.DamagePlayer(10);
                    StartCoroutine(Cooldown());
                    break;
                case (EnemyType.Ranged):
                    SoundScript.Playsound("enemyfire");
                    GameObject       bullet       =       Instantiate(bulletPrefab,
firePoint.position, Quaternion.identity) as GameObject;

bullet.GetComponent<enemybulletController>().GetPlayer(player.transform);
                    bullet.GetComponent<Rigidbody2D>().gravityScale = 0f;
                    bullet.GetComponent<enemybulletController>().isEnemyBullet      =
true;
                    //bullet.GetComponent<bulletController>().speed                 =
your_desired_speedf;
                    StartCoroutine(Cooldown());
                    break;
            }
        }
    }
    private IEnumerator Cooldown()
    {
        cdAttack = true;
        yield return new WaitForSeconds(cd);
        cdAttack = false;
    }
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.tag == "wall")
        {
            currState = EnemyState.Wander;
        }
    }
    public static void speedUpgrade(bool value)
    {
        //speed += 1;
        speedUPDATE = value;
    }
    public static void hitCountUp(int value)
    {
        //speed += 1;
        hittoDead += value;

    }

}
```

Player shooting script

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

public class playershooting : MonoBehaviour
{
    public Transform firePoint;
    public GameObject bulletPrefab;
    public GameObject shield;
    bool alreadySpawned = false;

    public float bulletForce = 20f;

    public int shootCount = 0;
    public int shootCD = 50;
    bool canShoot = false;
    public GameObject Reloading;
    public bool unlimitedshoot = false;


    private static bool unlimited = false;
    public static bool UnlimitedShot { get => unlimited; set => unlimited = value; }


    void Start()
    {
        Reloading.SetActive(false);
        shield.SetActive(false);

    }
    // Update is called once per frame
    void Update()
    {


        if (Input.GetButtonDown("Fire1") && canShoot == true)
        {
            Shoot();
            shootCount = shootCount + 1;

            if (shootCount == 5)
            {
                canShoot = false;
                shootCD = 50;
                StartCoroutine(shootCountReset());
            }
        }

        if (UnlimitedShot==true)
        {
            unlimitedshoot = true;
        }


        if (unlimitedshoot==true)
        {
            if (Input.GetButton("Fire1") )
            {
```

```
                Shoot();

            }
        }



        if (Input.GetKey(KeyCode.E))
        {
            StartCoroutine(shootCountReset());
        }

        if (Input.GetKey(KeyCode.Q))
        {
            if (alreadySpawned==false)
            {
                spawnShield();
            }

        }

    }

    void FixedUpdate()
    {




        if (shootCD == 0)
        {
            canShoot = true;
        }
        else
        {
            shootCD--;
        }
    }

    IEnumerator shootCountReset()
    {
        Reloading.SetActive(true);
        yield return new WaitForSeconds(2f);
        shootCount = 0;
        Reloading.SetActive(false);
    }

    void Shoot()
    {
        SoundScript.Playsound("fire");
        GameObject    bullet    =    Instantiate(bulletPrefab,    firePoint.position,
firePoint.rotation);
        Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();
        rb.AddForce(firePoint.up * bulletForce, ForceMode2D.Impulse);
        Destroy(bullet, 2f);
    }


    public void spawnShield()
    {
        StartCoroutine(shieldspawned());
    }

    IEnumerator shieldspawned()
    {
```

```
        //GameObject    shield    =    Instantiate(shieldPrefab,    firePoint.position,
firePoint.rotation);
        shield.SetActive(true);
        alreadySpawned = true;
        yield return new WaitForSeconds(3f);
        //Destroy(shield);
        shield.SetActive(false);
        alreadySpawned = false;
    }

    public static void setUnlimited(bool value)
    {
        UnlimitedShot = value;
    }
}
```

Lava damage script

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

public class lavaDamager : MonoBehaviour
{
    bool playerEntered;
    bool steppedIN;




    // Update is called once per frame
    void Update()
    {


    }


    private void OnTriggerEnter2D(Collider2D collision)
    {

        if (collision.tag=="Player")
        {
            playerEntered = true;
            Debug.Log("player step on lava");
            steppedIN = true;
            InvokeRepeating("dmgOvertime", 0f, 2f);

        }

    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.tag == "Player")
        {
            playerEntered = false;
            Debug.Log("player step out ");
            steppedIN = false;

        }

    }
```

```
    void dmgOvertime()
    {

        if (steppedIN==true)
        {
            GameController.DamagePlayer(1);
            GameController.isHurted = true;
        }
        else
        {
            CancelInvoke();
        }

    }

}
```

Heal player script

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

public class Heal : MonoBehaviour
{
    public GameObject healpoint;

     void OnTriggerEnter2D(Collider2D col)
    {
        if (col.tag=="Player")
        {
            GameController.HealPlayer(10);
            Destroy(healpoint);
        }
    }
}
```

Player shooting script

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

public class playershooting : MonoBehaviour
{
    public Transform firePoint;
    public GameObject bulletPrefab;
    public GameObject shield;
    bool alreadySpawned = false;

    public float bulletForce = 20f;

    public int shootCount = 0;
    public int shootCD = 50;
    bool canShoot = false;
    public GameObject Reloading;
    public bool unlimitedshoot = false;


    private static bool unlimited = false;
    public static bool UnlimitedShot { get => unlimited; set => unlimited = value; }


    void Start()
    {
        Reloading.SetActive(false);
        shield.SetActive(false);

    }
    // Update is called once per frame
    void Update()
    {


        if (Input.GetButtonDown("Fire1") && canShoot == true)
        {
            Shoot();
            shootCount = shootCount + 1;

            if (shootCount == 5)
            {
                canShoot = false;
                shootCD = 50;
                StartCoroutine(shootCountReset());
            }
        }

        if (UnlimitedShot==true)
        {
            unlimitedshoot = true;
        }


        if (unlimitedshoot==true)
        {
            if (Input.GetButton("Fire1") )
```

```
            {
                Shoot();

            }
        }



        if (Input.GetKey(KeyCode.E))
        {
            StartCoroutine(shootCountReset());
        }

        if (Input.GetKey(KeyCode.Q))
        {
            if (alreadySpawned==false)
            {
                spawnShield();
            }

        }

    }

    void FixedUpdate()
    {



        if (shootCD == 0)
        {
            canShoot = true;
        }
        else
        {
            shootCD--;
        }
    }

    IEnumerator shootCountReset()
    {
        Reloading.SetActive(true);
        yield return new WaitForSeconds(2f);
        shootCount = 0;
        Reloading.SetActive(false);
    }

    void Shoot()
    {
        SoundScript.Playsound("fire");
        GameObject    bullet    =    Instantiate(bulletPrefab,    firePoint.position,
firePoint.rotation);
        Rigidbody2D rb = bullet.GetComponent<Rigidbody2D>();
        rb.AddForce(firePoint.up * bulletForce, ForceMode2D.Impulse);
        Destroy(bullet, 2f);
    }


    public void spawnShield()
    {
        StartCoroutine(shieldspawned());
    }

    IEnumerator shieldspawned()
    {
```

```
        //GameObject    shield    =    Instantiate(shieldPrefab,    firePoint.position,
firePoint.rotation);
        shield.SetActive(true);
        alreadySpawned = true;
        yield return new WaitForSeconds(3f);
        //Destroy(shield);
        shield.SetActive(false);
        alreadySpawned = false;
    }

    public static void setUnlimited(bool value)
    {
        UnlimitedShot = value;
    }
}
```

Next level collider script

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

public class NextRoom : MonoBehaviour
{
    public GameObject entertxt,escapetxt;
    public int roomCount=0;
    bool gateCollide = false;
    public bool escapeStair;
    public bool lvl1,lvl2,lvl3;
    bool enteredr2, enteredr3 , enteredr4;


    public List<int> availableScenes;
    public List<int> playedScenes;

    // Start is called before the first frame update
    void Start()
    {
        entertxt.SetActive(false);
        escapetxt.SetActive(false);
    }

    // Update is called once per frame
    void Update()
    {
        if (gateCollide==true)
        {
            if (escapeStair == true)
            {

                if (Input.GetKey(KeyCode.E))
                {
                    entertxt.SetActive(false);

                    escapetxt.SetActive(true);


                }
            }
```

```
            if (escapeStair == false)
            {
                if (Input.GetKey(KeyCode.E))
                {
                    entertxt.SetActive(false);
                    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex
+ 1);
                }


    void OnTriggerEnter2D(Collider2D collision)
    {
        entertxt.SetActive(true);
        gateCollide = true;
    }

    void OnTriggerExit2D(Collider2D collision)
    {
        entertxt.SetActive(false);
        gateCollide = false;
    }
}
```

APPENDIX C: Functionality test data

Functionality test

Dungeon Revive developer: MUHAMMAD AMZAR RAIF BIN AMIR RASID

Date: 26/8/2021

No.	ID	Input (Question)	Expected Output (Answer)	Output (OK/Error/ Failed)
1	Menu.play button	Is the button functioning?	Move to next scene	OK
2	Menu.about button	Is the button functioning?	Text about the game	OK
3	Menu.quit button	Is the button functioning?	Application quit	OK
4	Player controls	Can player move or shoot?	Player can move, shoot, reload dash and shield	OK
5	Escape button	Can player pause the game?	Pause UI appear	OK
6	Interact.panels collider	Is a panel appear on the screen?	A panel showing controls, shop and achievement	OK
7	Shop.upgrade button	Can player upgrade a skill?	Skill upgraded and coin is reduced	OK
8	Interact.door	Player can interact with the door?	Player press button and move to next scene	OK
9	Shoot enemy	Can player kill enemy?	Player shoot enemy multiple time until the enemy dead	OK
10	HealthUI.take damage	Health bar reduced?	Player health is reduced	OK

Functionality test

Tester developer: NUR RIEZMAN NAIM BIN ZAMRI

Date: 27/8/2021

No.	ID	Input (Question)	Expected Output (Answer)	Output (OK/Error/ Failed)
1	Menu.play button	Is the button functioning?	Move to next scene	OK
2	Menu.about button	Is the button functioning?	Text about the game	OK
3	Menu.quit button	Is the button functioning?	Application quit	OK
4	Player controls	Can player move or shoot?	Player can move, shoot, reload dash and shield	OK
5	Escape button	Can player pause the game?	Pause UI appear	OK
6	Interact.panels collider	Is a panel appear on the screen?	A panel showing controls, shop and achievement	OK
7	Shop.upgrade button	Can player upgrade a skill?	Skill upgraded and coin is reduced	OK
8	Interact.door	Player can interact with the door?	Player press button and move to next scene	OK
9	Shoot enemy	Can player kill enemy?	Player shoot enemy multiple time until the enemy dead	OK
10	HealthUI.take damage	Health bar reduced?	Player health is reduced	OK