HO3

FYP Progress Report

2D Shooting Game

by
Chan Foo Him, Himi
Hon Tsz Pang, Andy
Lee Kwan Hang, Joe

Advised by
Prof. Andrew B. HORNER

Submitted in partial fulfillment
of the requirements for CPEG 4901
in the
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

2022-2023

1. Introduction

1.1 Overview

With the rise of hardware performance, games' graphic quality has become more and more realistic. 3D games have now become mainstream in the game industry. Many games are aiming at reaching the pinnacle of graphical quality. Game companies polish their game graphics, and put a large amount of budget on light calculation, object texture, and character modeling. However, they seem to neglect the real nature of games, the gameplay. We, as a player, are rightfully delighted due to the high graphical quality. At the same time, shooting games had become the most popular game among teenagers in 2021[1], with about 60%, which predicted that shooting games might become the primary game type in the future. Yet, First Person Shooting games, such as Counter-Strike: Global Offensive, and Apex Legend, have dominated the shooting game market nowadays. The game is an arena, which is player vs player(PvP), but not player vs environment(PvE). Besides, there are very few 2D shooting games being published compared to 3D shooting games due to the old-fashioned and unitary gaming mode. But, there is no doubt that shooting games are a popular game type in the recent trend. Therefore, 2D shooting games still have their own market value.

Recently, an outdated game, Project Zomboid, a 2D shooting and survival multiplayer sandbox game, has become a hot spot again. There are several reasons that it becomes popular again because of the shortage of this type of game, and players bored with the same type of game, i.e MOBA game, and Arena game. With this in mind, we are going to create a traditional 2D shooting game, combining the latest gameplay strategy.

1.2 Objective

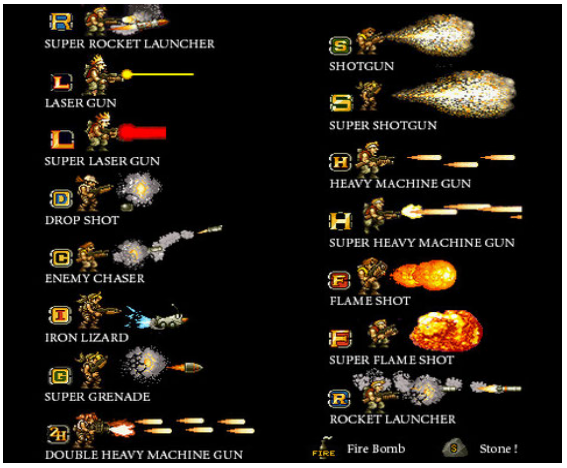
The target of this project is to create a 2D shooting game. We are developing the game through Unity, and we want to emphasize a satisfactory game for each player by combining most of the popular functions in some of the popular games, such as the crafting system, random map generating, high degree of freedom, and so on. In this project, we aimed to achieve a variety of goals:

1. Develop a 2D shooting game using Unity
2. The high degree of freedom in crafting weapons and playstyles.
3. Design an impressive main storyline
4. Create a variety of monster attack modes and skill sets.

1.3 Literature Survey

There are several classical 2D shooting games illustrating features and strategies worth learning and emulating to enhance the gaming experience.

1.3.1 Metal Slug



List of weapons

Rocket launcher

Fire gun

Metal Slug, a long-running 2D run-and-gun game series, was created by Nazca Corporation in 1996. It provides a large array of weapons. The player who is firstly equipped with a basic pistol can upgrade their weapons and unlock other guns ranging from shotguns, machine guns, and rocket launchers [3]. Each weapon has its shooting effects such as explosion effects, bullet trajectory, and rumble effects. Rather than shooting objects, it also emphasizes the importance of exploding objects and how enemies react with different weapons to improve the feeling of shooting.

Similar to Metal Slug, we would also like to increase the diversity of the weapon system which is one of the cores of a shooting game. With a wide range of weapons, players are

given more freedom of switching between weapons to deal with different enemies based on their gaming strategies. Moreover, we will also design the interactions of individual weapons with each enemy.

1.3.2 Enter the Gungeon

Enter the Gungeon is a bullet hell shooting game created by Devolver Digital in 2016[4]. Instead of normal bullet eruption, enemies in this game will mainly emit an endless wave of bullets which requires the player to predict and dodge the projectiles while aiming to attack the targets concurrently. Besides, the game map in this game is randomly generated each time the player enters the dungeon. The technique combines several small maps randomly to generate a unique map. Therefore, we can edit several small regions to achieve a hundred or thousand maps for the game.

In our project, we will apply a similar technique by entering the Gungeon.

First, regarding the monster attack technique, we would like to design several attack styles and attack patterns for each monster type. The second is the random map technique. By this technique, we can produce numerous maps for the player to maintain freshness.

1.3.3 Vampire Survivor

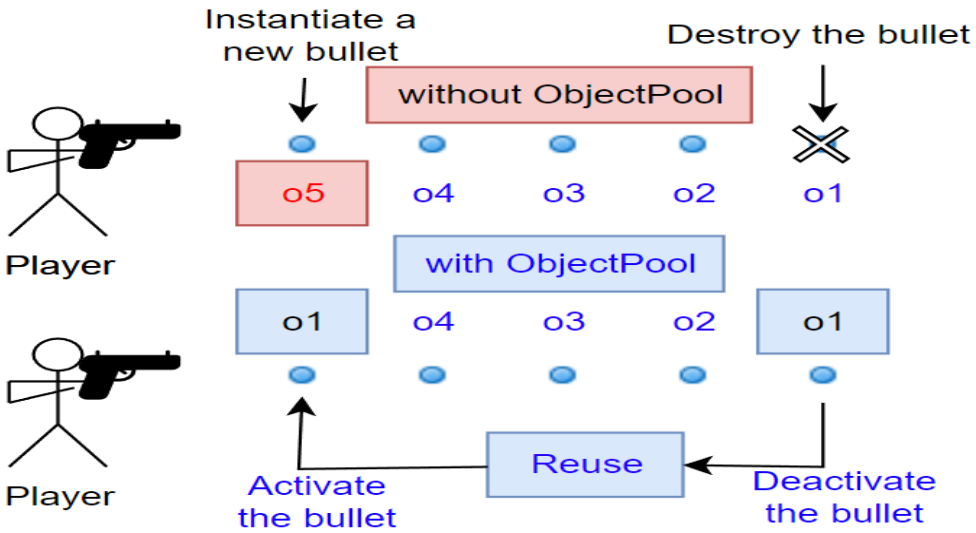
Vampire Survivor, a bullet hell-shooting game published by Indie Developer Poncle in 2021[5]. It is a survival game, which is only ended when the player dies. Besides, the monster will become stronger over time. Also, there will be supplies when the player kills monsters.

In our project, we would like to take advantage of Vampire Survivor, like the survival game mode, and the supply system. Since we are developing a shooting game, it will lack competitiveness if the ammunition is infinite. Therefore, we would like to have ammunition drops during killing the monsters or looting in the maps.

1.4 Technical Challenge

As a shooting game, the challenge we will face is that the high frequency of instantiating and destroying shooting and exploding objects may largely occupy the CPU processing power. For example, when the Player and the Enemy shoot at each other, bullets continue to be instantiated. When a bullet collides with other objects, the bullet is destroyed and an explosion effect game object is generated at the location.

To boost the effectiveness, we will utilize the ObjectPool approach [5] which provides an alternative and efficient strategy to handle these repetitive tasks. Shooting objects will be stored in a pool for reuse. If a request for the shooting object is received, the system will check for the availability of the previously created object. If there is an available object, it will be activated and returned. Otherwise, a new object will be created. Rather than destroying the object, it will deactivate and reset the object for reuse. With the object pool, the performance can be optimized by reducing the time for garbage collection and object creation.



2. Methodology

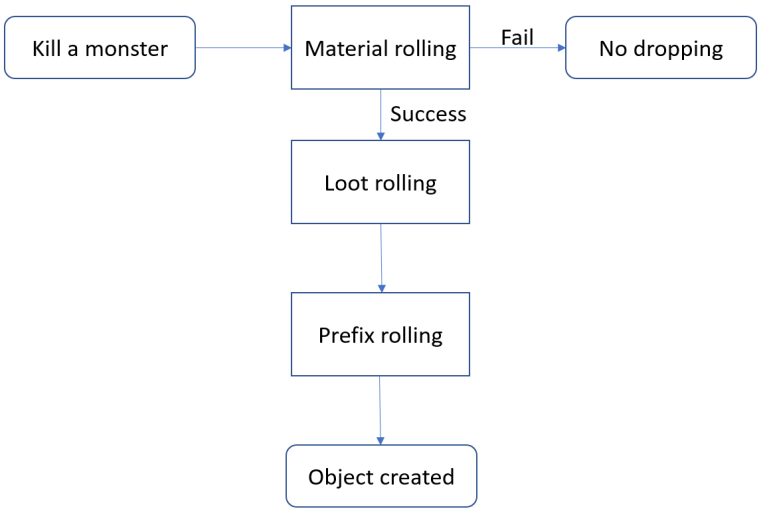
2.1 Design

2.1.1 Design the game story

We created the primary plot of our game, which can be followed throughout the game. Interesting storylines will entice people to play in an addicting manner.

2.1.2 Design the item (loot) dropping system

We designed the monsters to drop materials that are used for crafting or upgrading weapons with different additional prefixes items, which will be mentioned in the “weapon system” part. We created a distinct dropping probability for each of the prefixes.



2.1.3 Design the weapon system

We designed several weapon classes with individual animation and shooting patterns which are one of the main factors in enhancing the gaming experience of a shooting game. For the weapon system, we will have multiple ways to get weapons ranging from collecting materials to craft and killing the elite monsters respectively. The attributes of the weapons can be upgraded by the prefix materials.

The prefixes will include:

Prefix	**Effect**
Powerful	Increase DAMAGE
Speedy	Increase SPEED
Solid	Increase HEALTH POINT
Holy	Increase all attributes

2.1.4 Design the sound system

We designed the usage of several sound effects and background music for different weapons and game scenes respectively. The change of the music may enhance the immersion of the player, especially in the Boss Fight scene.

2.1.5 Design the monster type, movement, and behavior

We designed special features for the monsters. When the monster hits the players, negative buffs will be added to them. Such as bleeding, burns, and freezing. Each of them will have different effects.

Debuffs	**Effect**
Bleeding	Forbidden the Player’s dodge action
Burns	Continuously decrease health point
Freezing	Decrease moving speed

Besides, we would like to have 3 different levels of monster. They are normal, elite, and boss respectively.

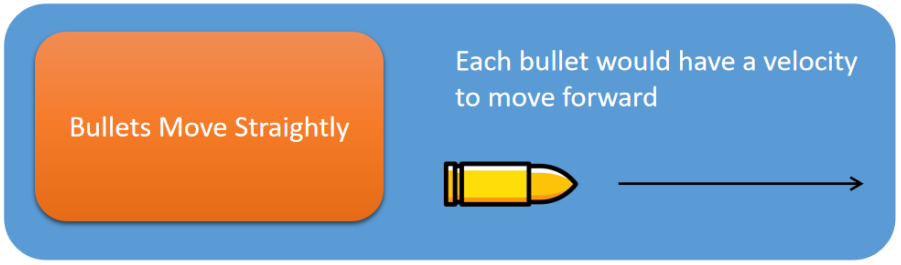
For the normal monster, their movement and skill are fixed. There will only be one attack style and a fixed walking speed. The elite monster, will have dynamic moving speed and a larger chasing range so that they may dodge the player’s attack and chase the player from a longer distance. Moreover, the boss will have at least two attack styles, such as attacking with a single fireball when their health is above 50% while keeping shooting fireballs in a radial pattern.

2.1.5 Design the shooting patterns

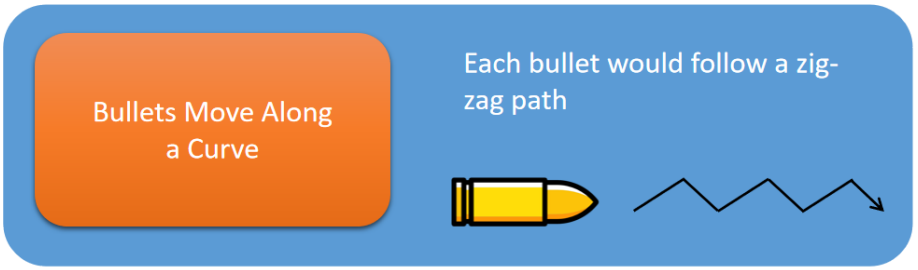
We designed the bullet moving patterns and divided them into 3 basic categories.

1. Bullets Move Straightly

   Each bullet moves in a straight line.

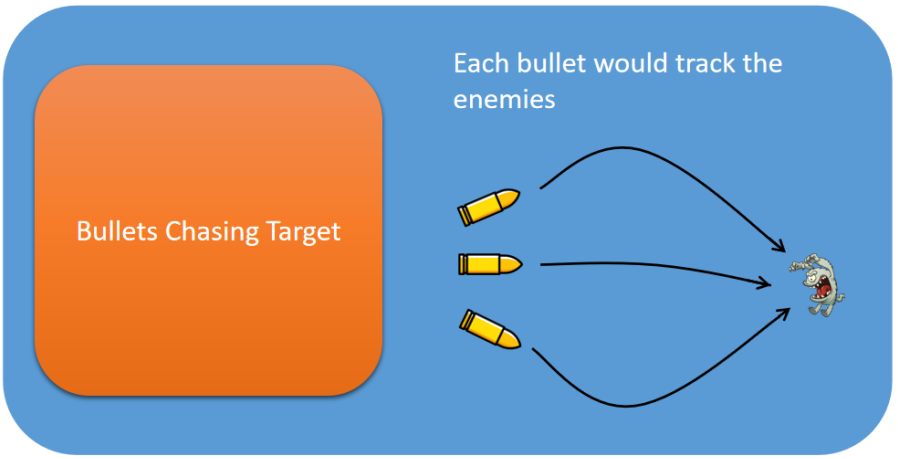


2. Bullets Move Along a Curve

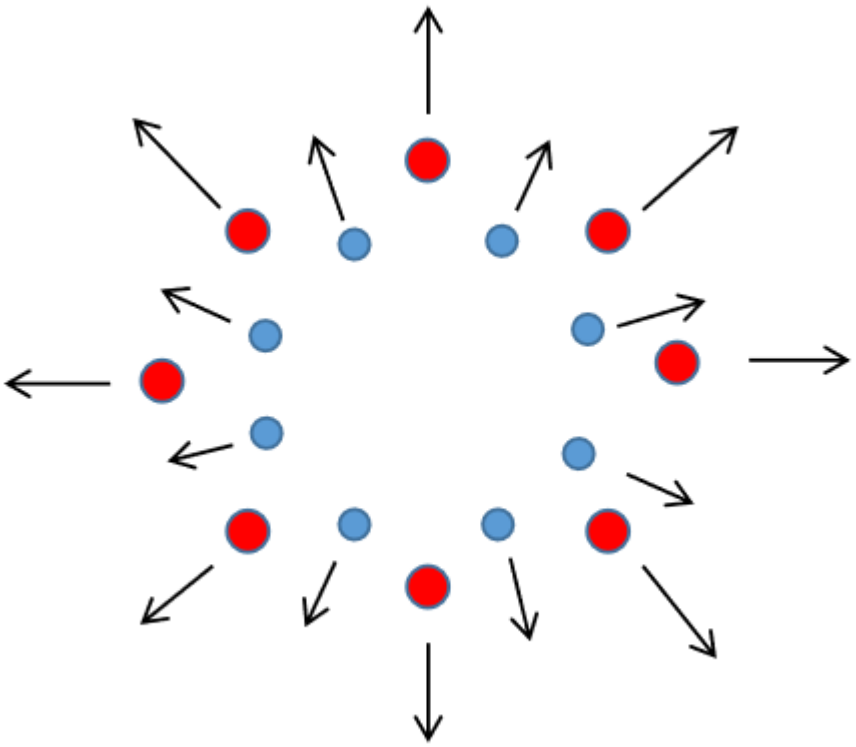


3. Bullets Chasing Target

   Each frame computes the direction between the bullet position and the target position. The interpolation value between the bullet's current direction and the previously determined direction is then calculated. Change the direction of the bullet to match the interpolation value.

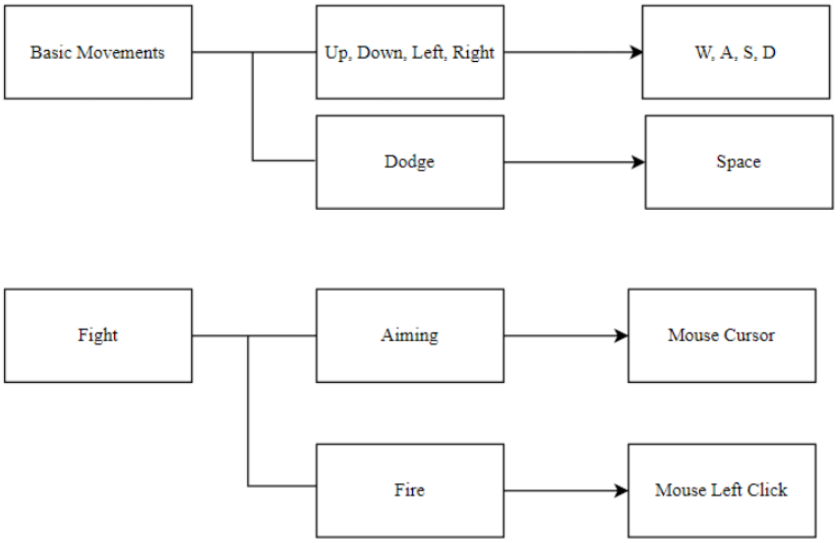


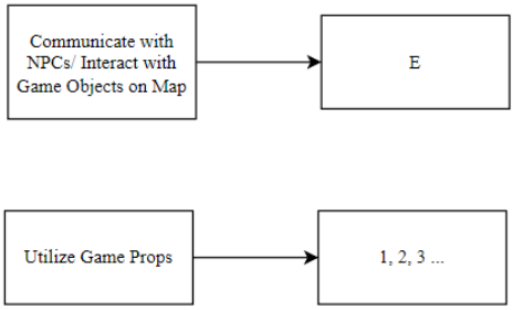
On top of these categories, the factors of bullet quantity, bullet spread, and shooting direction may provide a variety of different shooting patterns. A radial firing pattern, for example, can be created by composing the first bullet moving pattern (straight bullet movement) and assigning several shooting places to the game object to shoot bullets at the same time.



2.1.6 Design the gameplay mechanism

We designed a control system for the user to control the movements of characters and interact with the game objects on the map based on the inputs of mouse and keyboard buttons.





2.1.7 Design the mission types of the game

We designed two different mission kinds. They are the boss eliminating the mission and the hunting mission. Levels and game maps will change between missions.

1. **Boss eliminating mission**

   To kill the boss to complete the mission.

2. **Hunting mission**

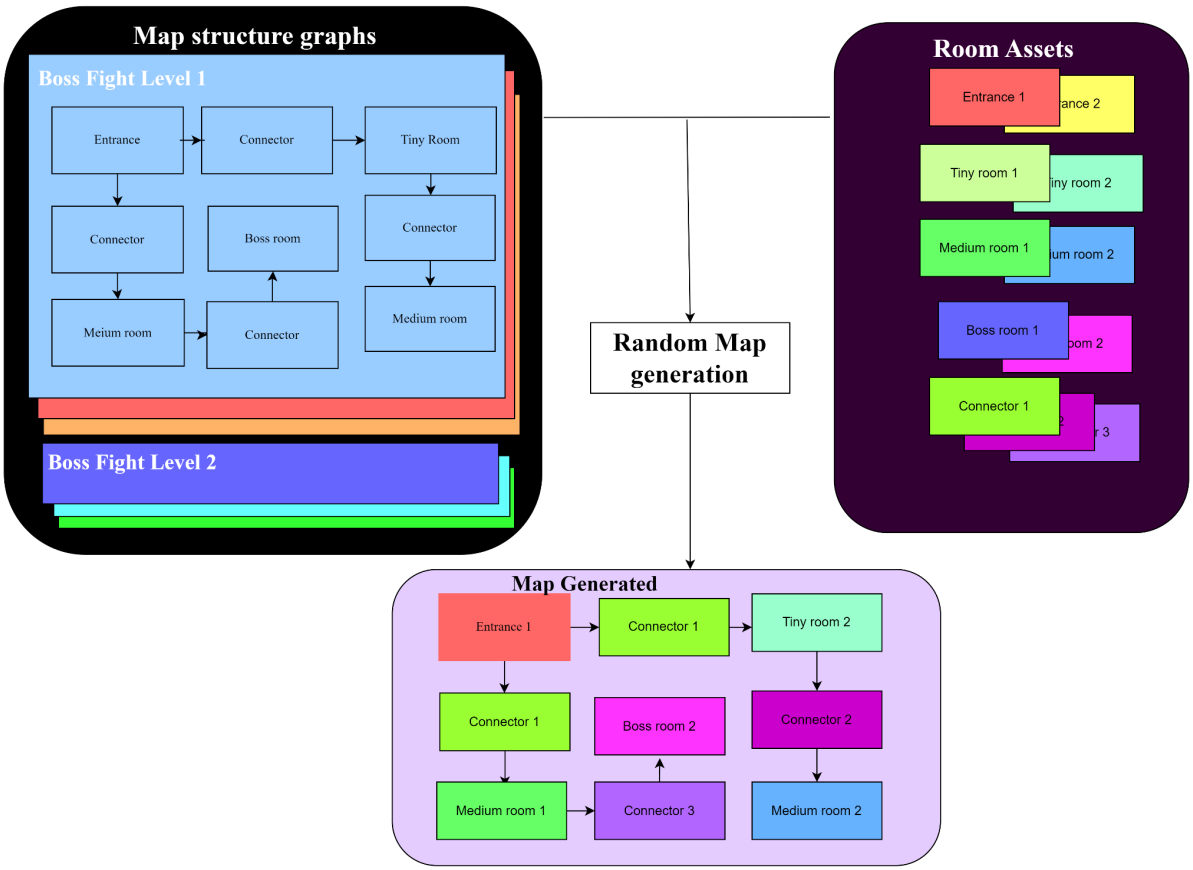
   To kill the specified amount of monsters within a time constraint to fulfill the mission.

2.1.8 Design a game map randomization process

We designed to divide the map randomization process into 3 modules.

Module 1 is to design multiple map structure graphs for different levels and missions to store the correlations of individual room blocks.
Module 2 is to design room assets for separate room types.
Module 3 is to randomly select a map structure graph as the blueprint and choose the room asset from the pools matching the room types specified by the map structure graph and connect them in a random direction.

After completing the module 1 and 2 manually, module 3 will combine them to generate a map each time the game is loaded, the map will have a high probability that it will not be the same as the one appeared in the previous gameplay.



2.1.8.1 Design the room types

We designed 7 types of rooms ranging from entrance room, tiny room, medium room, big room, connector room, chest room, and boss room respectively, which will become the fragments of the map. Different rooms have different features.

Room type	Description
Entrance	Starting location when the map is loaded
Tiny room	Contains 1-3 normal monsters
Medium room	Contains 2-4 normal monsters and 1 elite monster
Big room	Contains 5-6 normal monsters and 2-3 elite monsters

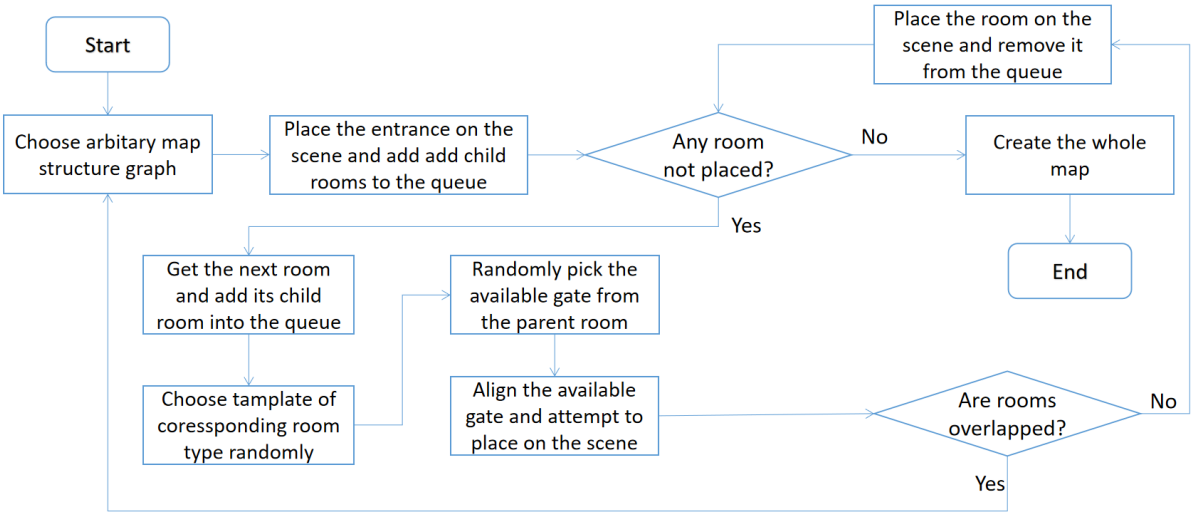
Chest room	Gives life points or other equipment to the Player.
Connector room	Act as a corridor to connect the rooms
Boss room	Contains one boss. The boss room should be locked until all the monsters in other areas are defeated.

2.1.8.2 Design the rule of map structure graph

We designed to store the correlations of individual room blocks as a map structure graph for map randomization. The map structure graph also keeps track of the parent and child rooms for each room block. For a boss eliminating mission, a valid graph should have one entrance and one boss room. For a hunting mission, a valid graph should have one entrance and one big room.

2.1.8.3 Design the map randomizing algorithm

We designed an algorithm to create a random map based on the predefined sets of map graphs and room assets. All rooms on the map structure graph contain an array of children rooms as mentioned. The current room and the child rooms will be added to a queue for processing.



2.1.9 Design the pathfinding component for enemy movement

We need to create our own pathfinding component for enemy movement because Unity does not contain one for 2D projects. To accomplish the pathfinding feature, we designed to use the A* search algorithm to work with the Unity tilemap grid. With the A* search algorithm, we can obtain the shortest path from the enemy location to the player location. This path will be used to control the enemy movement.

For the A* search algorithm, it calculates the cost of the surrounding cells from the current cell and moves to the neighboring cell with the lowest cost for the next iteration until the target cell is located. The total cost of a neighboring cell, denoted as f_cost, is the sum of g_cost (the distance from the parent cell to this cell) and h_cost (the distance from this cell to the target cell). Because the computation time is quite low, we will use Manhattan distance to determine h_cost.

2.2 Implementation

We are using Unity to construct the game. As for art resources and music resources, we decided to get them from the Internet such as Asset Store based on the copyright ordinance. When necessary, we create and modify some resources by ourselves. During the implementation, we are using Git for version control and GitHub as our remote repository.

2.2.1 Implement the gameplay mechanics

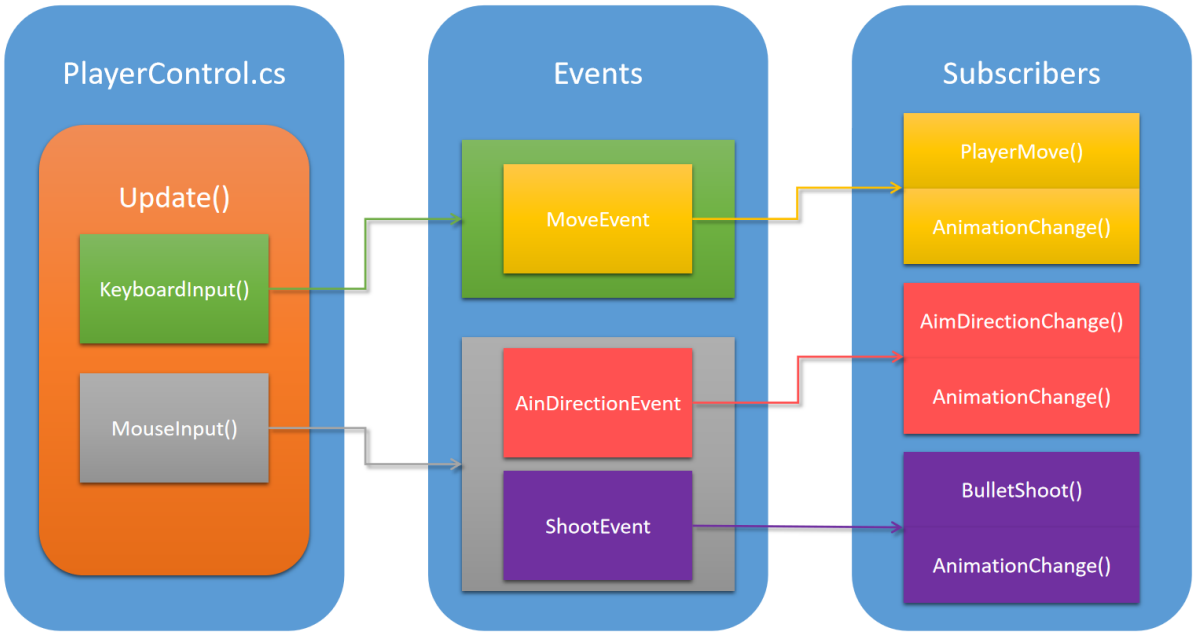
We implemented a PlayerControl script to receive and handle user input data at each frame such as mouse aiming direction and keyboard inputs. Various inputs were combined with relevant events.

MoveEvent: be triggered when the User inputs W, A, S, D.

AimDirectionEvent: be triggered when the User moves the mouse cursor.

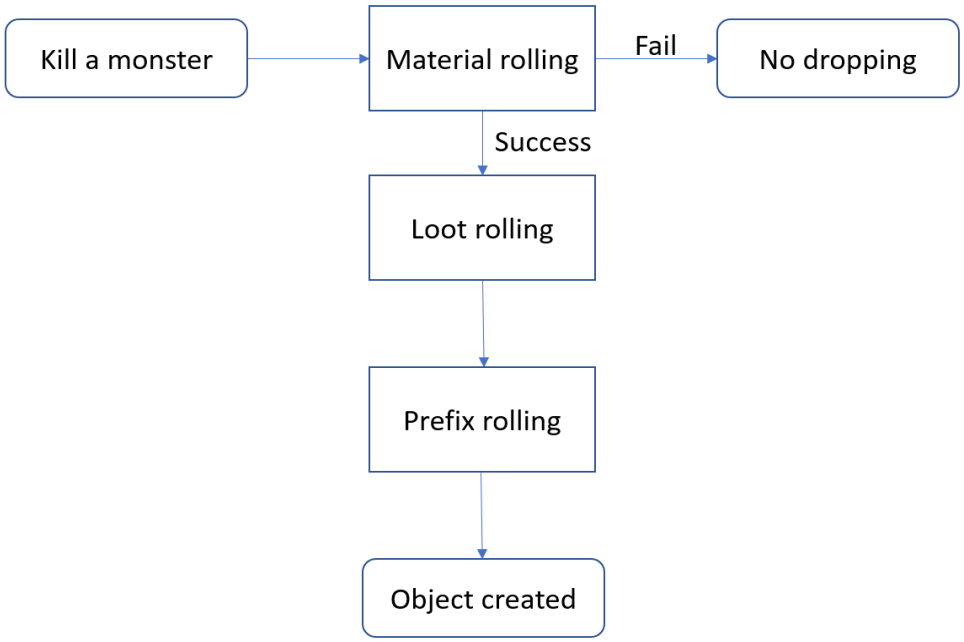
ShootEvent: be triggered when the User press Mouse Left Click

We adopted an observer design pattern to handle these events. With this pattern, we could efficiently hook the functionality of a script as a subscriber to the events. Once the event was triggered, the subscribers would be automatically invoked to handle the event. This design method greatly enhanced code reusability. MoveEvent, AimDirectionEvent, and ShootEvent were also utilized by enemy classes. However, subscribers to the events functioned differently.



2.2.2 Implement the item (loot) dropping system

We are implementing a loot dropping system. After killing a monster, there will be a drop formula. All drops go through several rolls to determine what the player will receive. The defeat of the monster will trigger an event for material rolling.



Other than the rolling, the rareness of the material drop will proportionally increase with respect to the monster type. For example, normal monsters will only drop normal quality material. However, elite monsters and boss monsters will only drop items at a higher rank.

2.2.3 Implement the weapon system

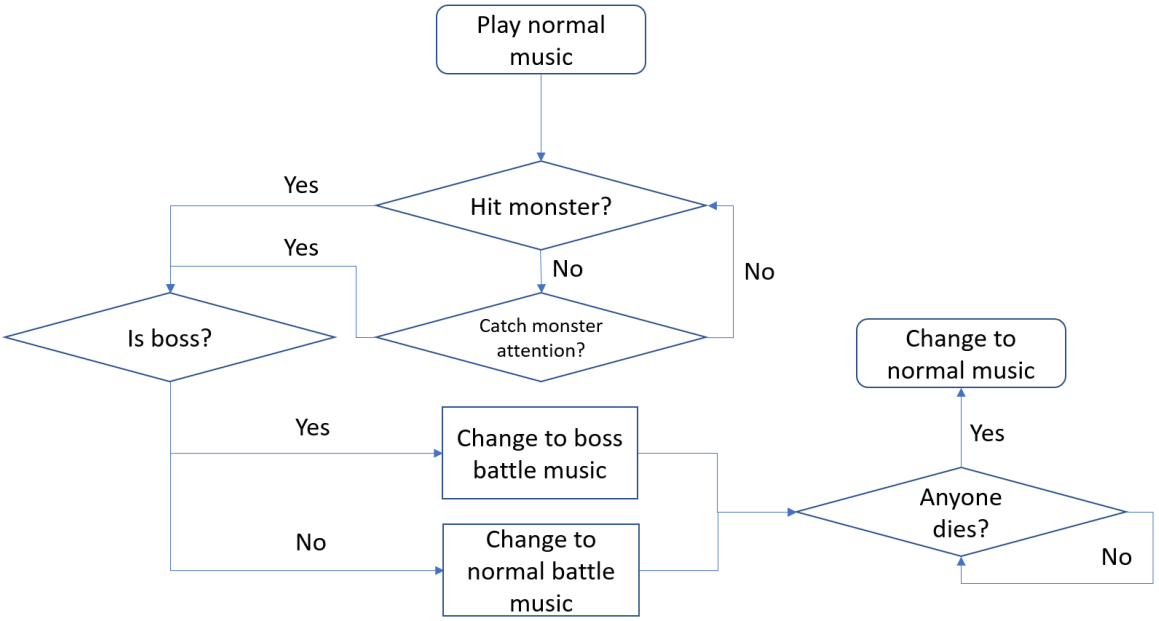
We implemented a ScriptableObject as a data container to store the attributes for different weapons.
Besides, we are going to group each type of the crafting material. No matter what prefixes they are, they belong to the same array so as to simplify the classification for the crafting system.
For each weapon, there will be a Bullet component controlling the shooting bullet ranging from updating the position in each frame, requesting a bullet object from ObjectPool manager, and triggering a corresponding event when the bullet collides with another game object.

2.2.4 Implement the sound system

We are developing a weaponSound script, when the events of shooting or hitting are triggered, the weapon sound effect clip should be accurately played.

For background music, we will classify kinds of music into three categories: normal music, normal battle music, and boss battle music. We will implement a background music manager to be executed throughout the gameplay. The music manager will handle the switch between background music when different events are triggered.

We will use Audacity, or other digital audio generation software to create different music.



2.2.5 Implement the monster type and movement

For different monster types, we created individual classes to store the status and debuff details.

Debuffs were divided into two groups - instant debuffs and long-term debuffs.

For instant debuffs, we just do simple math calculations to modify the player's status data.

For long-term debuffs, we will set a timer to constantly change the player's status value during the gameplay.

2.2.5.1 Implement the pathfinding component

We are implementing a Pathfinding.cs script to compute and return the route from the starting position to the destination by using A* search algorithm. A MonsterMove component is being developed to retrieve the suggested path from the Pathfinding script and control the movement of the monster.

For each monster, there will be a chasing range, once the Player is within the chasing range, the relative method of Pathfinding.cs will be invoked and the monster will move to the region surrounding the present location of the Player. Because we do not want the monster to go along the obstruction, we will apply weighting such that cells near the obstacle are less likely to be picked as the chasing path.

2.2.6 Implement the mission types of the game

Following the flow of the storyline, we will gradually release some hunting missions and boss-eliminating missions.

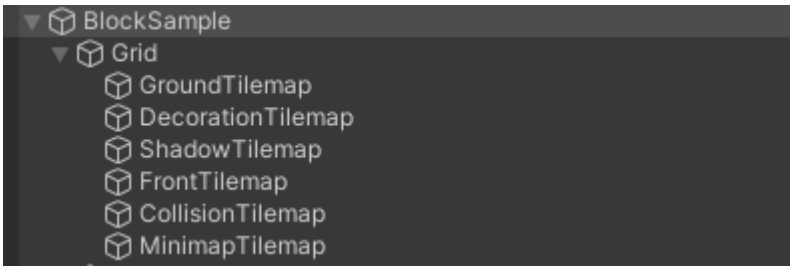
In the quest system, the boss eliminating missions will be automatically accepted according to the plot. If a player wants to accept hunting missions, the player needs to visit the mission board to get the mission details and accept them.

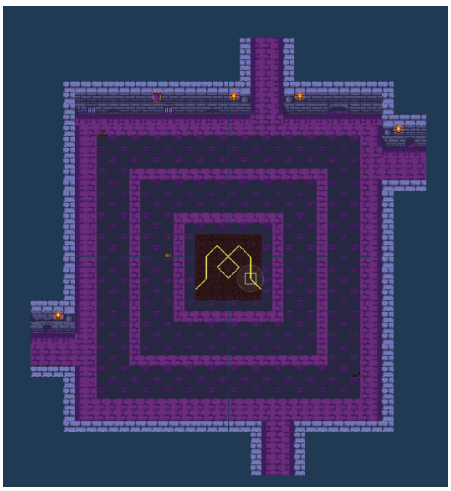
2.2.7 Implement the game map

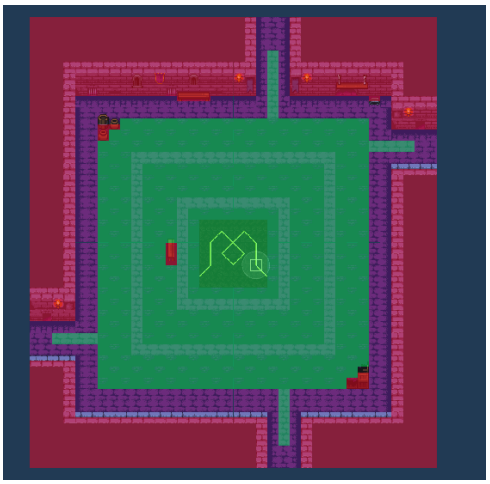
2.2.7.1 Implement the room assets

We made use of the built-in tilemap component of Unity to design a variety of room blocks for map randomization. For each room, we implemented a Room ScriptableObject to store the details of the room block such as room size, type, and gate locations.

For every single block, we created 7 layers, which were the ground layer, decoration layer, shadow layer, front layer, collision layer, and minimap layer respectively. Different layers perform different functionalities. Also, the priority of the layers is as follows in the picture below.







Ground Layer and Front Layer **Collision Layer**

Ground layer was used to place the wall and the floor for the room. Also, the player's model will always have a higher priority to display than the objects belonging to the ground layer.

Different from the Ground layer, the front layer was used to show the object over the player, with the player's character model being partially obscured by the object on the front layer. For instance, when the player is behind a wall, the wall object should partially cover the player's model.

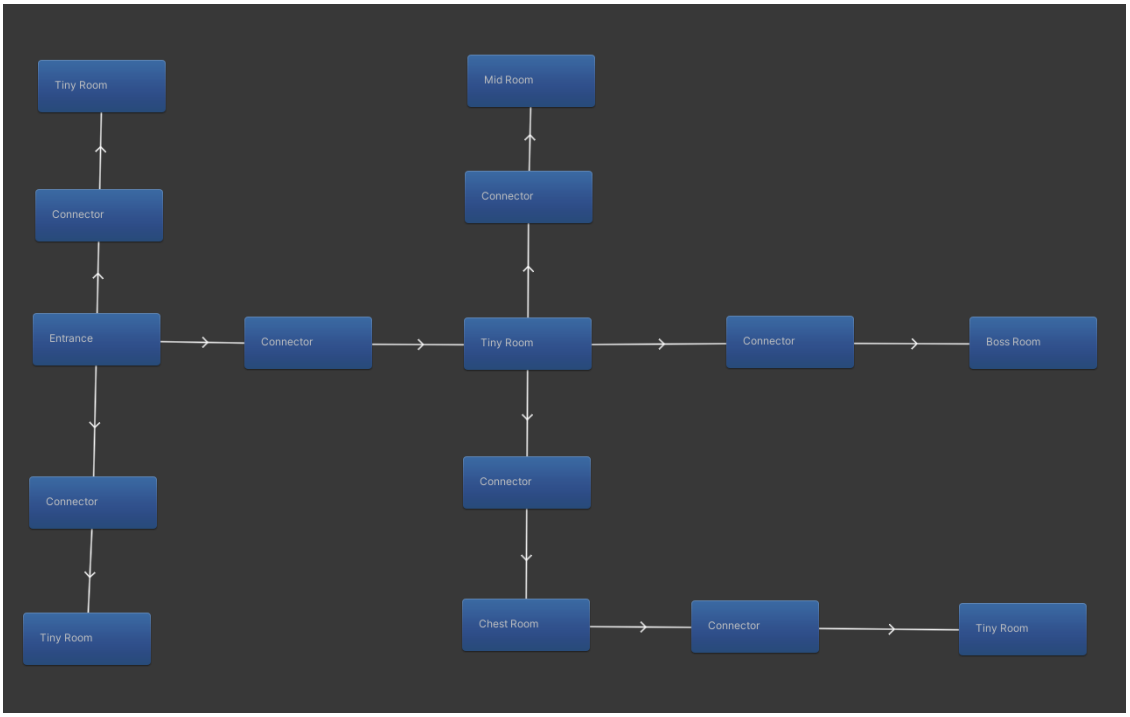
The Collision layer has two purposes. One option is to create a collider by using the Tilemap Collider 2D component to the wall to prevent the player and enemy from going through it. Another purpose is to mark preferred paths and obstacles on the map for Enemy pathfinding.

2.2.7.2 Implement the map structure graph

We created a custom map structure graph editor in Unity to help with the design process. The designer can construct several sorts of room blocks by right-clicking the editor window and

left-clicking any block to drag a connecting line to other room blocks. When the connection line is connected, the relationship between the parent and kid rooms is immediately saved.

We will implement a validation check on the map structure graph for different missions. At this stage, the designer can develop any graphs, and some of them should be considered invalid. For instance, the map for boss-eliminating missions does not include a boss fight room.



Valid map structure graph with boss eliminating objective

2.2.7.3 Implementing the game map generator

We implemented a map manager containing an array of map structure graphs for different missions and room assets details. The map manager will be used to handle the random generation of game maps according to the accepted mission. During gaming, the manager will maintain track of the player's progress to handle scene changes.

2.3 Testing

The game undergoes three testing modes, unit test, system test, and alpha test.

In unit tests, we tested some modules or systems separately. In this stage, we focus on the underlying logic of the game. That is to say, we just focus on whether the data process result reaches our expectations.

In the system test, we tested some interactions between modules and systems. In this stage, we focus on the business logic of the game. We will test whether the connection of APIs reaches our requirements, and polish them when necessary.

In the alpha test, we will invite others to perform a black box test. In this stage, we focus on the subjective feeling of players like the shooting feelings or smoothness of moving animation.

2.3.1 Test the item (loot) dropping system

We will test the dropping chance of loots. As this module involves randomness, we will generate massive test results to get a precise dropping chance.

2.3.2 Test the weapon system

We will control characters to shoot at targets with different weapons to test the weapon system. This progress can test whether damage calculation is correct and whether weapons are with suitable shooting range and pattern, as well as the bullet reloading.

2.3.3 Test the sound system

We will prepare different monsters on the map and let the player walk through those areas to test whether the background music is switched properly. We will specifically test some rare cases like player escape or monsters suddenly joining the battle when hunting bosses.

2.3.4 Test the monster type and movement

We will deliberately control the character to get hurt by the monster to test for those debuffs and damage created. We mainly focus on the data consistency with a combination of debuffs, or a combination with buffs and investigate whether there are conflicts.

To evaluate the dependability and stability, we will construct a route visualizer that converts the Pathfinding component's output into a visualized game object on the map.

2.3.5 Test the gameplay mechanics

We controlled the character to perform simple tests on all key events, and test whether those events reach our expectations especially when a single key can trigger multiple events.

2.3.6 Test the mission types of the game

We will test whether missions are correctly shown on the mission board and whether the mission can be correctly submitted. Besides, we need to test whether there are story conflicts between missions.

2.3.7 Test a randomized game map

We tested whether regions are correctly and randomly generated. We generated massive sample data to test whether the random algorithm gives us the correct expected value, and tested whether all regions are reachable. Moreover, we tested if the collision is correct so that the player cannot pass through it.

2.3.7.1 Test the room types

We tested whether all rooms are having correct room type classification. Besides, we are testing whether all rooms have the correct size and gates position so that rooms can be connected flawlessly.

2.3.7.2 Test the map structure graph

We tested whether the map structure graph editor correctly showed and modified the map graph. We also tested that every room should have the correct relation between the parent room and the child room.

2.3.7.3 Test the map randomizing algorithm

We tested whether the scenes are generated randomly according to the map structure graph and tested whether the algorithm can avoid generating overlapped scenes. Moreover, we will check whether the room and the connector are connected correctly according to the coordinate of the entry.

2.4 Evaluation

After finishing the alpha test, we will collect comments from testers to get the game experience. We mainly focus on the following field:

- Visual experience
    - Is the FPS high enough (Movement looks smooth)?
    - Are those artworks' styles good?
    - Is the animation smooth?
    - Does the shooting feel good and real?
    - Is the monster having enough distinct skill sets?
- Auditory experience
    - Is the music attractive and does it match the game?
    - Will the music distract you from playing?
- Gameplay
    - Is the game too easy, or too difficult?
    - Is the UI user-friendly?
    - Does the story well penetrate the whole game?

All the improvements revolve around the objectives.

3. Project Planning

3.1 Division of work

Design

Design Task	**Himi**	**Andy**	**Joe**
Design the story of the game	✓	✓	✓
Design the mission type of game	✓		✓
Design the item-dropping system	✓	✓	
Design weapon system	✓	✓	✓
Design the music system		✓	✓
Design the gameplay mechanics	✓	✓	
Design the game maps	✓	✓	✓

Implementation

Implementation Task	**Himi**	**Andy**	**Joe**
Implement the mission type of game	✓	✓	
Implement the item-dropping system		✓	✓
Implement the weapon system	✓	✓	✓
Implement the music system		✓	✓
Implement the gameplay mechanics	✓	✓	
Implement the game maps		✓	✓

Testing

Testing Task	Himi	Andy	Joe
Test the mission type of game	✓	✓	✓
Test the item-dropping system	✓	✓	✓
Test the weapon system	✓	✓	✓
Test the music system	✓	✓	✓
Test the gameplay mechanics	✓	✓	✓
Test the game map	✓	✓	✓
Test the whole system	✓	✓	✓

3.2 GANTT Chart

Design

Design Task	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Design the main story of the game	✓	✓	✓							
Design the mission type of game			✓							
Design the item-dropping system	✓	✓	✓							
Design weapon system		✓	✓	✓						
Design the music system				✓	✓	✓	✓			
Design the gameplay mechanics		✓	✓	✓	✓					
Design the inventory system					✓	✓	✓			
Design the game maps		✓	✓	✓	✓	✓	✓			

Implementation

Implementation Task	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Implement the mission type of game			✓	✓	✓					
Implement the item-dropping system							✓	✓	✓	
Implement the weapon system						✓	✓	✓	✓	
Implement the gameplay mechanics				✓	✓	✓	✓	✓		
Implement the music system							✓	✓	✓	✓
Implement the inventory system					✓	✓	✓	✓	✓	
Implement the game maps				✓	✓	✓	✓	✓		

Testing

Testing Task	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
Test the mission type of game								✓	✓	✓
Test the item-dropping system								✓	✓	✓
Test the weapon system								✓	✓	✓
Test the music system								✓	✓	✓
Test the gameplay mechanics								✓	✓	✓
Test the inventory system								✓	✓	✓
Test the game maps							✓	✓	✓	✓

4. Required Hardware and Software

4.1 Hardware

Recommended hardware requirements:

OS	Window 10 32-bit/64-bit
Processor	Intel Core i5-8500
Memory	8GB
Graphics	NVIDIA GeForce GTX 750
Storage	~4GB
Sound Card	16-bit stereo, 48kHz

4.2 Software

Unity	Game engine for including the map editing, game logic, sound effects
Git	Version control system to coordinate collaboration between members of the project
VScode	Companion to Unity for writing codes and debug
Audacity	Digital Audio Editor
C#	Programming Language

5 References

[1] Statista. “Most popular video game genres worldwide 2021, by age group.” https://www.statista.com/statistics/1263585/top-video-game-genres-worldwide-by-age/ [Accessed 3, September 2022]

[2] Lucas. “Metal Slug Review.” https://www.ign.com/articles/2008/05/30/metal-slug-review [Accessed 4, September 2022]

[3] Enter the Gungeon Official. “Enter the Gungeon.” https://dodgeroll.com/gungeon [Accessed 10 September 2022].

[4] Fandom. “Vampire Survivors Wiki.” https://vampire-survivors.fandom.com/wiki/Vampire_Survivors_Wiki **[**Accessed 12, September 2022**]**

[5] Unity Official."Introduction to Object Pooling - Unity Learn." https://learn.unity.com/tutorial/introduction-to-object-pooling# [Accessed 6 September 2022].

6 Appendix A: Meeting Minutes

6.1 Minutes of the 1st Project Meeting

Date: June 5, 2022
Time: 15:00
Place: Discord
Present: Himi, Andy, Joe
Absent:
Recorder: Himi

1. **Approval of minutes**

This was the first formal group meeting, thus none of the minutes to approve.

2. **Report on progress**

2.1 All of the team members have read the instructions of the FYP form and understood the requirements.
2.2 All the team members agreed to choose shooting as the main type of game.

3. **Discussion items**

3.1 Himi and Joe discussed which styles of maps (Mission per map or a single huge open world map) will be used.
3.2 All the team members have shared their ideas about the game features, and game stories.
3.3 Himi suggested adding a weapon crafting system allowing players to customize the strength and stats of their weapons.
3.4 Andy and Himi proposed implementing a random loot-dropping system for players to collect materials for crafting different weapons.

4. **Goals for the coming week**

4.1 Design to use either Unity or Unreal Engine for the project.

5. **Meeting adjournment and next meeting**

The meeting was adjourned at 18:00
The next meeting will be on July 9 and held on Discord.

6.2 Minutes of the 2nd Project Meeting

Date: July 9, 2022
Time: 14:00
Place: Discord
Present: Himi, Andy, Joe
Absent:
Recorder: Himi

1. **Approval of minutes**

The minutes of the last meeting were approved without amendment.

2. **Report on progress**

2.1 All of the team members agreed to use Unity for the FYP project.
2.2 All the team members share a main focus on the project.

3. **Discussion items**

3.1 All the team members discussed both Unity and Unreal Engine in different aspects, including the difficulty to learn the usage of the engines, the availability of different types of assets, and the popularity of the engines.
Seeking the consent of all members, we eventually selected to use Unity for our game development project.

3.2 All the team members have shared their ideas about the game features, and game stories.

4. **Goals for the coming week**

4.1 All the team members will look for some tutorials about the usage of Unity to gain some basic knowledge and understanding of it by the next meeting.

5. **Meeting adjournment and next meeting**

The meeting was adjourned at 15:30
The next meeting will be on August 30 and is expected to be held on Discord.

6.3 Minutes of the 3rd Project Meeting

Date: August 30, 2022
Time: 20:00
Place: Discord
Present: Himi, Andy, Joe
Absent:
Recorder: Himi

1. **Approval of minutes**

   The minutes of the last meeting were approved without amendment.

2. **Report on progress**

2.1 All members have gained essential pieces of knowledge of Unity.

3. **Discussion items**

3.1 Each member shares their idea on the distribution of work.
3.2 Each member designs which part they are responsible for.
3.3 For the proposal report, Himi will write the Overview and Objectives part. Andy will be responsible for the Literature survey and Design, and Joe will be in charge of the Implementation, Testing, and Evaluation parts.

4. **Goals for the coming week**

4.1 All the team members will start to design how to implement the tasks assigned to them according to the chart and share their progress.

5. **Meeting adjournment and next meeting**

The meeting was adjourned at 21:30.
The next meeting will be on September 20.

.

6.4 Minutes of the 4th Project Meeting

Date: September 20, 2022
Time: 17:45
Place: Discord
Present: Himi, Andy, Joe
Absent:
Recorder: Himi

1. **Approval of minutes**

   The minutes of the last meeting were approved without amendment.

2. **Report on progress**

2.1 Each member has barely shared how they will implement some of their tasks.

3. **Discussion items**

3.1 Andy suggested using random map generation techniques to make our game map.
3.2 Joe tried to make the UI interface inside Unity.
3.3 Himi designed to separate the map into several layer for providing a better game quality and more clear architecture when drawing it.

4. **Goals for the coming week**

4.1 All the members will try to implement Unity's interface for the map builder.
4.2 All the members will start to find the assets for the map, such as the wall, the floor, as well as the decoration, and the shadow.

5. **Meeting adjournment and next meeting**

The meeting was adjourned at 19:15.
The next meeting will be on October 21.

6.5 Minutes of the 5th Project Meeting

Date: October 21, 2022
Time: 19:45
Place: Discord
Present: Himi, Andy, Joe
Absent:
Recorder: Himi

1. **Approval of minutes**

   The minutes of the last meeting were approved without amendment.

2. **Report on progress**

2.1 Every member tried to make the map builder's UI interface but there are some bugs that needs to be solved.
2.2 Andy found some of the assets for the map block.

3. **Discussion items**

3.1 Joe designs to have at least 3 levels for the map, which means that more blocks will be placed if the level of the map is higher.
3.2 Himi designs to have several block tilemap template prefabs for each of the part of the map block so as to achieve the random map generation technique.

4. **Goals for the coming week**

4.1 Himi starts to make an overview architecture for the map.
4.2 Andy starts to implement the player movement as well as the control system
4.3 Joe starts to implement the lighting system.

5. **Meeting adjournment and next meeting**

The meeting was adjourned at 21:00.
The next meeting will be on November 13.

6.6 Minutes of the 6th Project Meeting

Date: November 13, 2022
Time: 14:00
Place: Discord
Present: Himi, Andy, Joe
Absent:
Recorder: Himi

1. **Approval of minutes**

   The minutes of the last meeting were approved without amendment.

2. **Report on progress**

2.1 Barely finished the random map generation algorithm while there is still some of the bugs.
2.2 Finished the map lighting system.
2.3 Found the player's model and edited the player's dodging animation texture.
2.4 Implemented the control system of the player's movement.

3. **Discussion items**

3.1 Andy suggests to have only 2 mission types, which is boss elimination and hunting mission.
3.2 Joe designs to have different rareness of material dropping, which the rareness drop will increase proportionally with the ranking of the monster. (i.e normal, elite, and boss type)
3.3 Himi suggests separating the room template based on the current level. (i.e level 1 will have different sets of room template when compare with level 2)

4. **Goals for the coming week**

4.1 Fixes the bugs in the random map generation algorithm.
4.2 Correct all the incorrect layers put in each room template.
4.3 Try to implement a basic weapon crafting system (i.e without any prefix)

5. **Meeting adjournment and next meeting**

The meeting was adjourned at 15:30.
The next meeting will be on December 15.

6.7 Minutes of the 7th Project Meeting

Date: December 15, 2022
Time: 17:30
Place: Discord
Present: Himi, Andy, Joe
Absent:
Recorder: Himi

1. **Approval of minutes**

   The minutes of the last meeting were approved without amendment.

2. **Report on progress**

2.1 The bug of the random map generation algorithm is solved by setting up the coordinate of each door.
2.2 Most of the map is displayed correctly for the player's model.

3. **Discussion items**

3.1 Andy designs to have different shooting patterns per each type of gun.
3.2 Design how to implement the monster movements and attack styles.

4. **Goals for the coming week**

4.1 Find out the texture for monsters and implement the animation for them.
4.2 Try to understand and implement the AStar Algorithm
4.3 Adding special effects for the bullet shooting.

5. **Meeting adjournment and next meeting**

The meeting was adjourned at 19:00.
The next meeting will be on Jan 17.

6.8 Minutes of the 8th Project Meeting

Date: Jan 17, 2023
Time: 20:00
Place: Discord
Present: Himi, Andy, Joe
Absent:
Recorder: Himi

1. **Approval of minutes**

   The minutes of the last meeting were approved without amendment.

2. **Report on progress**

2.1 Added some special effects on the bullet, such as spark and explosion.
2.2 Created static movement and attack style for the monster.

3. **Discussion items**

3.1 Andy suggests making the shooting become more real by adding recoiling. (especially for the shotgun)
3.2 Everyone designs whether utility(grenade) should be added or not.

4. **Goals for the coming week**

4.1 Adding sound effects for shooting.
4.2 Implement the damage calculation for each type of monster.
4.3 Complete the weapon combination system that accepts different rareness of material.
4.4 Try to implement some movable objects to the game map.

5. **Meeting adjournment and next meeting**

The meeting was adjourned at 21:00.
The next meeting will be on February 21.