HO3

FYP Proposal

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

Table of Contents

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 game, 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 carried out 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, had dominated the shooting game market nowadays. The game kind 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 game is 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Recently, an outdated game, Project Zomboid, a 2D shooting and survival multiplayer sandbox game, has become a hot spot again.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:

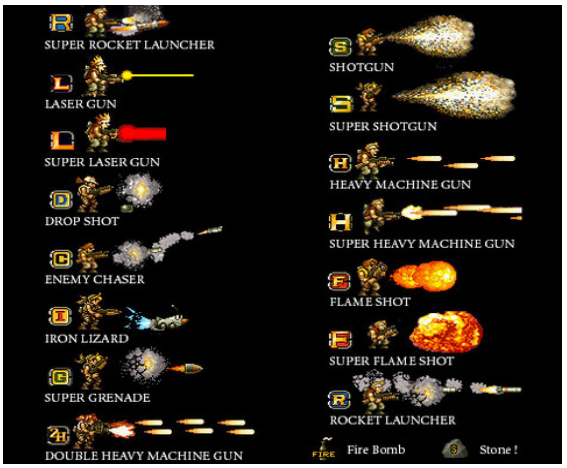
1. Develop a 2D shooting game using Unity
2. The high degree of freedom in crafting weapons and playstyles.
3. Design an impressive main storyline and several interesting side stories
4. Create a variety of monster attack modes and skill sets.

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. To boost the effectiveness, we will utilize the Official ObjectPool API offered by Unity [2] providing 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 keep the object for reuse. With the usage of an object pool, the performance can be optimized by reducing time for garbage collection.

1.3 Literature Survey

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

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 short guns, 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.

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.

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 monsters are killed by the player.

In our project, we would like to take the 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. We, on the other hand, are going to have a survival game mode similar to it.

2 Methodology

2.1 Design

2.1.1 Design the game story

We have designed the main storyline of our game, and this story can go through the whole game. Besides, we are going to design some side stories to increase the savor of it if the time is on our schedule.

2.1.2 Design the character upgrade system

We first designed the character to upgrade by gaining experience from killing the monsters or completing missions. The level will limit the weapons that the player can equip. Also, the player will gain basic stats when leveling up, such as HP, moving speed, and stamina for running.

2.1.3 Design the item (loot) dropping system

We first designed the monsters to drop materials that are used for crafting weapons. Besides, the materials will have different additional prefixes, which will be mentioned in the “weapon system part”. We have designed to have different dropping probabilities for each of the prefixes. Moreover, we will add a special sound effect when the drop is rare.

2.1.4 Design the weapon system

We will design several weapons with individual animation and shooting effects which are one of the main factors in enhancing the gaming experience of a shooting game. For the weapon system, we design to have multiple ways to get weapons ranging from buying from the

NPCs, collecting materials to craft, and killing the elite monsters respectively. The power of the weapon depends on the rarity of the materials.

In addition, for the materials, we would design to add some special bonus stats. For example, if we use a “speedy” rod to craft a weapon, it will give extra firing speed for the weapon.

At present, the prefixes include:

Prefix	**Effect**
Powerful	Increase ATK
Speedy	Increase SPD
Solid	Increase DEF
Holy	Increase ATK, SPD, DEF

Rarity of weapons	**Border color**
Crude	White
Fair	Blue
Exquisite	Purple

Peerless	Gold

2.1.5 Design the sound system

We will design 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. In addition, we will have special sound effects for rare materials dropping.

2.1.6 Design the type of monsters

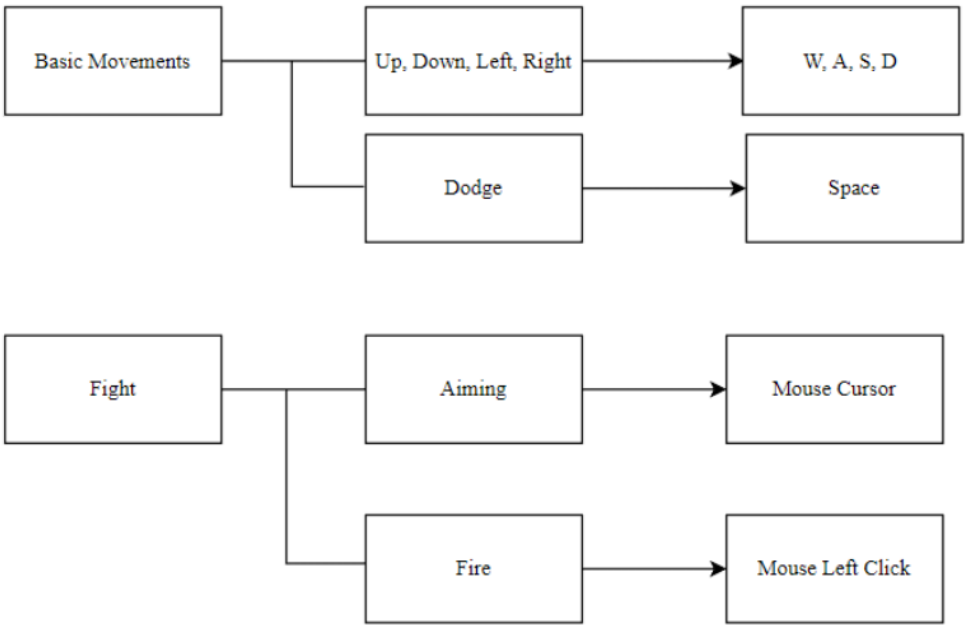
We have designed to add special features to the monsters, for example, when the monster hits the players, negative buffs will be added to them. Such as bleeding, radiation, scorch, frostbite, poison, and tinnitus. Each of them will have different effects, and the only way to counter those negative effects is to use the relative medical items or get medicines from the nurse NPC.

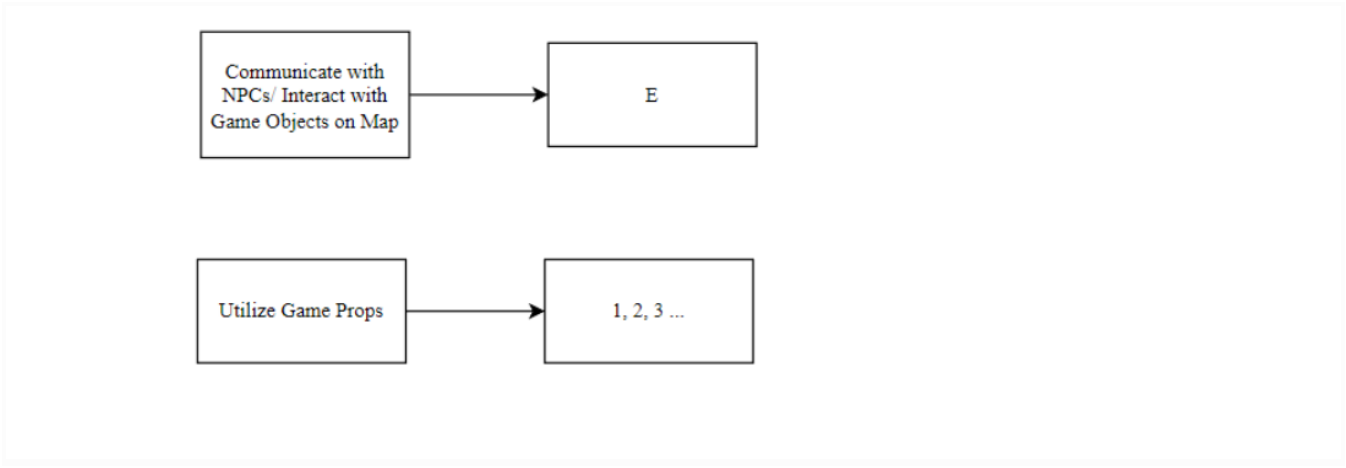
Debuff	Effect
Bleeding	Decrease the HP per minute and walking speed
Radiation	Decrease the HP per minute and
Scorch	Take extra damage and decrease the aiming accuracy
Frostbite	Take extra damage and decrease the stamina for running
Poison	Decrease the HP per minute(the decrease amount increases as the time increases)
Tinnitus	Loss of hearing sense

Other than that, there will be special monsters, like elite monsters and bosses. We are going to have different attack abilities and styles for each of them. Also, the higher the difficulty of the mission, the more smooth the action of monsters will be.

2.1.7 Design the gameplay mechanics

We will design 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.





2.1.8 Design the mission types of the game

For the mission types, we will design to have 4 types. They are survival missions, hunting missions, collecting missions, and rescue missions to increase the diversity of the game. And thus, increasing the taste of the game.

1. **Survival mission**

   To survive for several minutes, like 10 minutes or 15 minutes, to finish the mission.

2. **Hunting mission**

   To kill the specified amount of monsters to fulfill the mission.

3. **Collecting mission**

   To collect the specified items to achieve the mission.

4. **Rescue mission**

   To help the special NPC to accomplish the mission.

2.1.9 Design the game maps

We will design to have several regions or rooms as the blueprint to use a simple random map generation algorithm to connect different regions to make the map to be unique every time.

2.2 Implementation

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

2.2.1 Implement the game story

The progression of the plot relies on the dialogue system between characters, so we will plot a table or a tree diagram for storing dialogue data, including dialogue ID, character, content, and the next dialogue's ID.

	A	B	C	D	E	F	G
1	ID	Character	Content	Next	branch	effect	target
2	0	NPC_A	Hello	1	0		
3	1	Player	Hi	2	0		
4	2	NPC_A	How are you	3	0		
5	3	Player	I am fine	5	1	HP@1	Player
6	4	Player	No so good···	6	1	HP@-1	Player
7	5	NPC_A	Nice to hear that	-1	0		
8	6	NPC_A	Oh, sorry to hear that	-1	0		

```
using SystemAcl.Colletions;
using SystemException.Collections.Generic;
using UnityEngine;

public class DialoManager : MonoBehaviour {
    // Dialogue file in csv
    public TextAsset dialogDataFile;

    // Sprite image
    public Sprite sprite;

    // character name
    public TextMeshPro name;

    // content
    public TextMeshPro dialogText

    private void Start() {
        UpdateText("NPC A", "This is the first sentence of the game")
    }

    private void Update() {

    }

    public void UpdateText(string name, string text) {
        name.text = name;
        dialogText.text = text;
    }

    //TODO: read csv file
}
```

2.2.2 Implement the character upgrade system

We will create a class to store the player's status and carry out value calculations. For example, when the player carries a damage buff so that the damage created is doubled, we will get the final damage value to monsters by a formula like

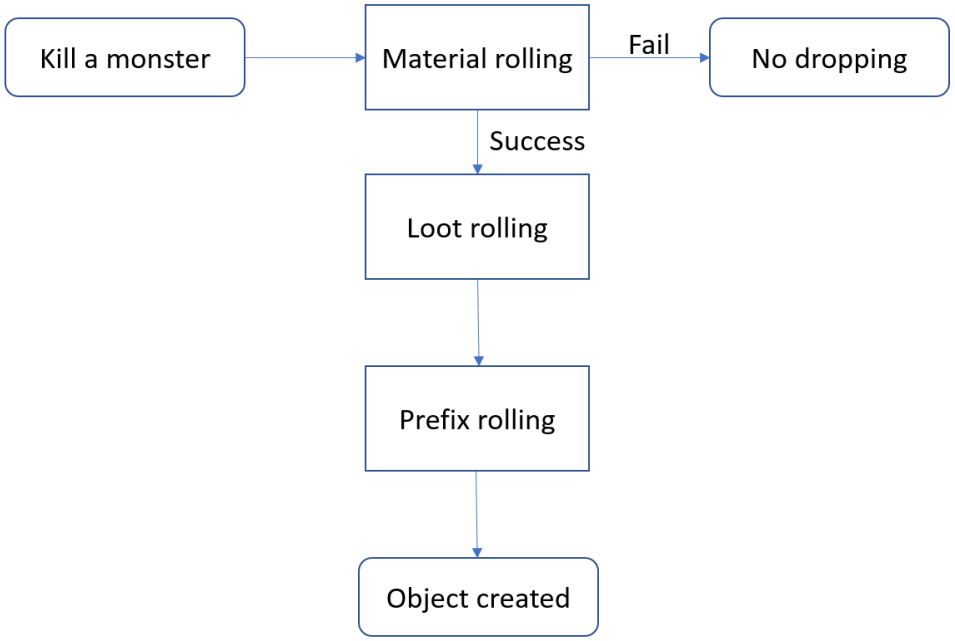
$$damage = player's\ attack\ power\ *\ buff\ coefficient\ -\ monster's\ defense\ power$$

This upgrade system is not only for modifying damage value but also for defense value, moving speed, etc.

When a player changes equipment, level judgment is carried out. If the player's level is not enough, the player cannot equip that equipment.

2.2.3 Implement the item-dropping system

After killing a monster, there is a drop formula. All drops go through several rolls to determine what the player will receive.



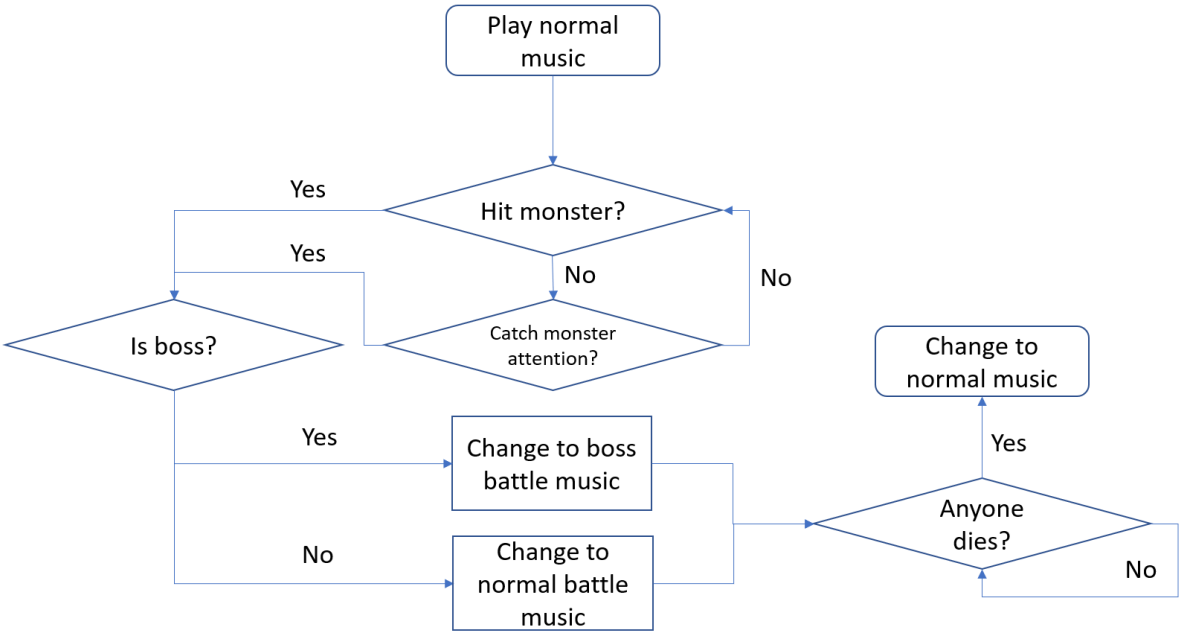
2.2.4 Implement the weapon system

We will implement different weapons. Besides the variety of damage, we also have other attributes like shooting range, shooting speed, etc.

2.2.5 Implement the sound system

We may use Audacity, or other digital audio generation software to create different music. For weapons' sound effects, we will trigger the playing of sound effect audio clips when shooting and hitting.

For background music, we will classify kinds of music into three categories: normal music, normal battle music, and boss battle music.



2.2.6 Implement the type of monsters

Debuffs are 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.

2.2.7 Implement the gameplay mechanics

We will bind those keys to corresponding events. A single key event will just trigger a single event, e.g., if NPCs are standing together, the player presses ‘E’ to communicate with the closest one rather than talking to both. Loot drop picking is no exception.

If there are event conflicts, for example, when the player presses ‘W’ and ‘S’, the character will remain stationary. With the same logic, the player press ‘W’ and ‘D’, and the character will move toward the top right.

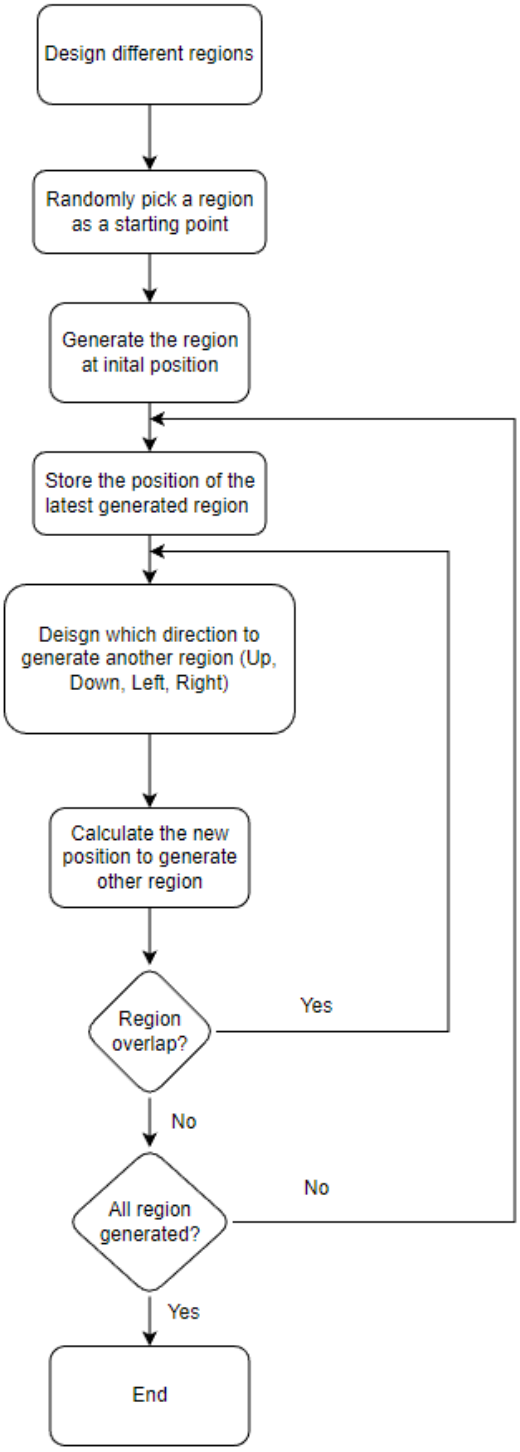
2.2.8 Implement the mission types of the game

Following the flow of the storyline (main quest), we will gradually release some side missions, including but not limited to collecting missions, hunting missions, and so on (refer to “Design the mission types of game”).

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

2.2.9 Implement the game maps

We first have several predefined regions. When creating a game map, we will randomly define the starting point. Then, we will randomly choose a direction each time after a region is created until all regions are generated.



2.3 Testing

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

In unit tests, we will test each module or system 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 will test the interaction 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 game story

To test the game story, we will scan through each dialogue to check the flow of each dialogue and the corresponding images. Besides the word spelling and grammar, we will also test the player's interaction with the dialogue.

2.3.2 Test the character upgrade system

We will prepare test cases to test the limitation due to level restriction and the result of status value change. For example, we will test whether level restriction can restrict the players to equip specific equipment and whether the calculation is correct after status values are changed.

2.3.3 Test the item-dropping system

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

2.3.4 Test the weapon system

To test the weapon system, we will control characters to shoot at targets with different weapons. This progress can test whether damage calculation is correct and whether weapons are with suitable shooting range and frequency.

2.3.5 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.

2.3.6 Test the type of monsters

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.

2.3.7 Test the gameplay mechanics

We will control 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.

2.3.8 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.

2.3.9 Test the game maps

We will test whether regions are correctly and randomly generated. We will generate massive sample data to test whether the random algorithm gives us the correct expected value, and test whether all regions are reachable.

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?
- Auditory experience
    - Is the music annoying?
    - Will the music distract you from playing?
- Gameplay
    - Is the game too easy, or too difficult?
    - Is the UI user-friendly?

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

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

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

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