Dungeon Hunter

1. <u>Project Overview</u>

This is a 2D action game developed by using Pygame. Players need to shoot enemies, pass all levels, and defeat a boss. Players can use money that they get from shooting enemies to buy a weapon or upgrade it to make it easier to kill enemies.

2. <u>Project Review</u>

This game is 2D action game that has gameplay like “Soul Knight” .This game also has enemies and boss that hard to defeat and have combat moves that depends on player movement and action like “Dark Souls”

3. <u>Programming Development</u>

<u>3.1 Game Concept</u>

Players must clear all levels by fighting enemies and defeating a boss that can predict the player's movement.
-Movement: walking, dashing
-Combat system: Shooting, Using magic skills, and special abilities
-Enemies: Different enemies types that have different behaviours
-Shop and power-ups: Can buy items from shops or collect items to enhance a gameplay
-Stat tracking: tracking death count, score, time, money, and shooting accuracy

<u>3.2 Object-Oriented Programming Implementation</u>

1.Player Class

-Attributes: health, speed, dash range, move direction, weapon
-methods: move(), attack(), dash(), draw(), take_damage()

2.Enemy Class

-Attributes: health, speed, damage, move direction

-methods: move(), attack(), draw(), take_damage()

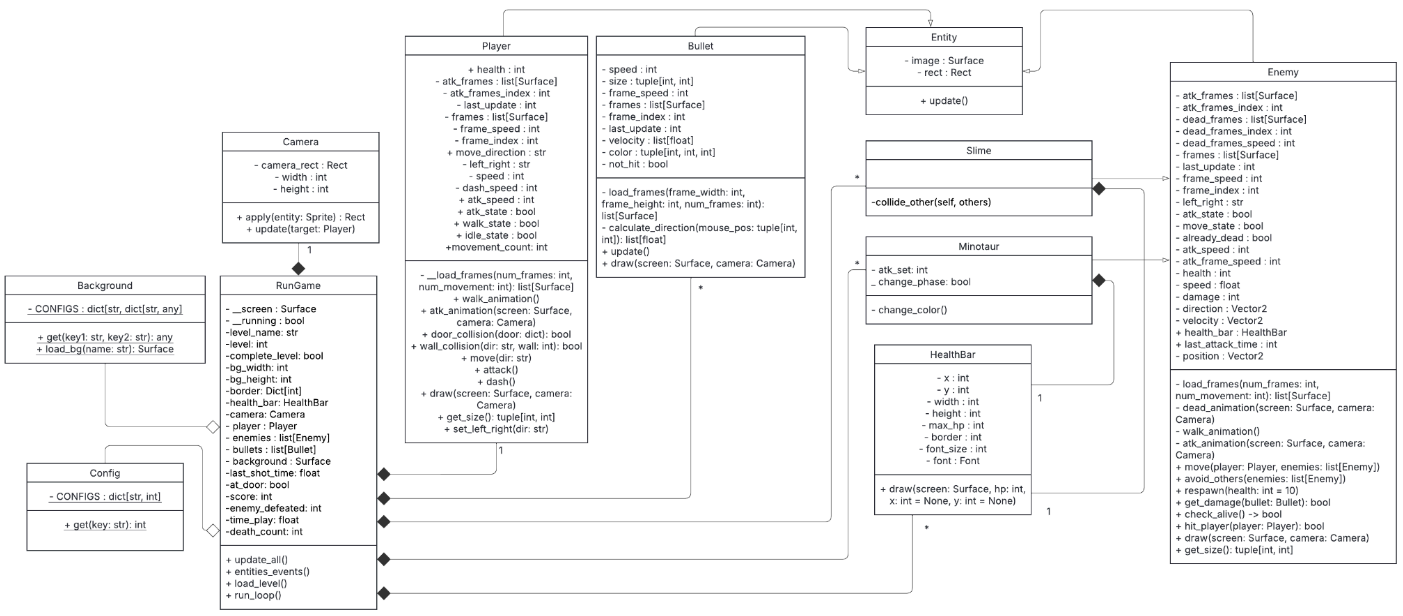
3.Bullet Class
-Attributes: location, speed, color, radius
-methods: calculate_direction(), hit_enemies(), draw()

4.Boss Class
-Attributes: health, speed, damage, move direction, attack_move_set
-methods: move(), attack(), draw(), take_damage(), extra_attack()

5.Run_game Class
-Attributes: screen, background, player, bullets
-methods: update_all(),run_loop(),pause(),restart(),start(), track_stat()

6.UI Class
-Attributes: health_bar, score, timer, shop
-methods: update_ui(), draw()

7.Config Class
-Attributes: CONFIGS contain value that use in a game



-methods: get()

UML LINK:
https://lucid.app/lucidchart/4d0d02c8-7a70-41a0-a514-ec49460c037e/edit?viewport_loc=-252%2C-245%2C2299%2C1001%2CHWEp-vi-RSFO&invitationId=inv_721849d6-78e0-4955-99e8-566e74bfdfb5

3.3 Algorithms Involved

-**Collision**: if players collide with enemies, players will lose HP
-**Enemies behaviour**: enemies will track the player location then move in a straight line to the player(using Pythagorean theorem).
-**Boss behaviour**: Boss will track the player's behaviour.
For example:
-if players use skills that they can't move for a while the boss will use skill to attack players
-the boss can dodge player attacks by tracking the location of player bullets. Boss will move sideways to dodge bullets if the distance of bullets is near the boss(use distance formula to calculate distance).
-**Event Driven**: Keypress handling movement and attack
-**Randomization**: enemies randomly spawn

4. Statistical Data (Prop Stats)

4.1 Data Features

1.Player movements: distance that move, dash used
2.Death count: count player's death in a whole game
3.Time: Time that spent in level and a whole game
4.Enemy interact: Enemies that defeated, amount of attack that hit them
5.Score: level that complete, scores

4.2 Data Recording Method

The data will be stored in the CSV file.

4.3 Data Analysis Report

-Use graph to show death count, scores, and enemy interact compare to time taken
-Use table to show numbers of items that player collected and player movement

	Why is it good to have this data? What can it be used for	How will you obtain 50 values of this feature data	Which variable and which class will you collect this from?	How will you display this feature data (via summarization statistics or via graph)?
Player movements	To know that distance in levels is short or long to reach new level	Collect total movement every 30 seconds	movement_count variable in Player class	Using histogram
Survived time	To balance a difficult level of the game	Collect duration time when player die	survived_time variable in RunGame class	Using histogram
Enemy defeated every 1 minute	To balance a game and know that enemy should be stronger or not	Collect total enemy defeated every 1 minute	enemy_defeated variable in RunGame class	Using histogram
Score every 1 minute	To balance a game and can compare score with other players	Collect score that player get every 1 minute	score variable in RunGame class	Using histogram
Item that player buy	To make items in game have balance and know what items should be adjust	Collect when player buy items	bought_list in shop class	Using histogram

Enemy type that defeated	To balance a game and know that which type of enemy should be stronger or not	Collect when enemy get defeated	enemy_defeated_type in RunGame class	Using histogram
Time that use in level	To balance a time that use in level	Collect when player complete each level	level_completed dict in RunGame class. key is level and value is time	Using histogram
Level that player complete	To know that which level is hard	Collect when player complete each level	level_completed dict in RunGame class. key is level and value is time	Using histogram

Table

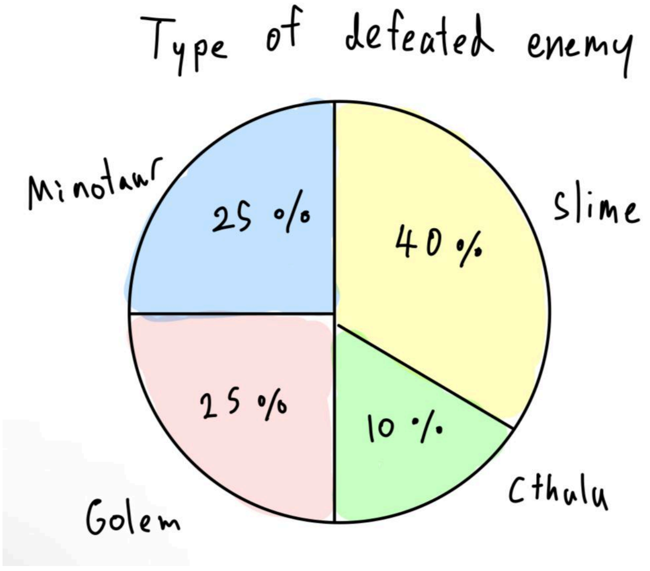
Play Times	Level complete	Mean of enemy defeated every 1 minutes	Mean of score every 1 minutes	Amount Item that player buy in that round	Survived time	Mean of player movement every 30 minutes
1						
2						

Graphs

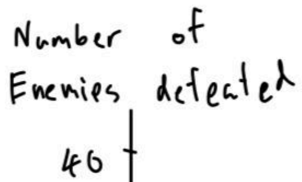
	Feature Name	Graph Objective	Graph Type	X-axis	Y-axis
Graph 1	Type of enemy that defeated portion	To show that player defeated what enemy the most	Pie chart	None	None
Graph 2	Enemy	To analyze	Histogram	Time (1	Number of

	Defeated Per minute	the trend of enemy defeat rate over time		minute intervals)	Enemies Defeated
Graph 3	Level that player complete	To show how far players progressed in the game	Line graph	levels	Frequency of player that complete
Graph 4	Score that get in 1 minute	To show that how much score do player get in 1 minute	Box plot	Score that get in 1 minute	None
Graph 5	Items Purchased by Players	To show that which items purchased by Players	Bar Graph	Items that player purchased	frequency

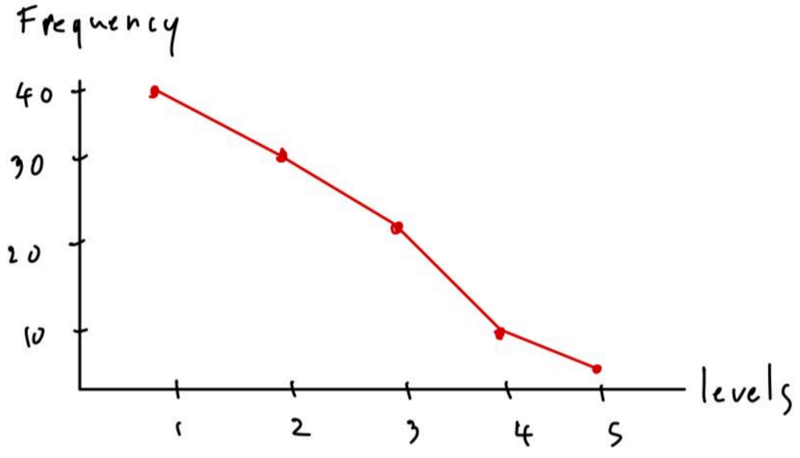
Graph1



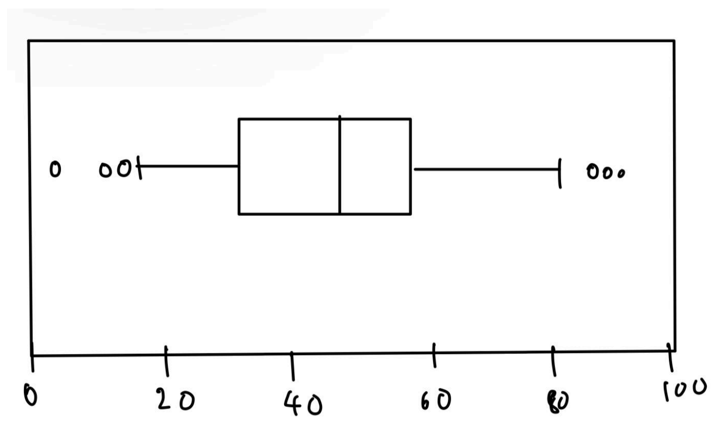
Graph 2



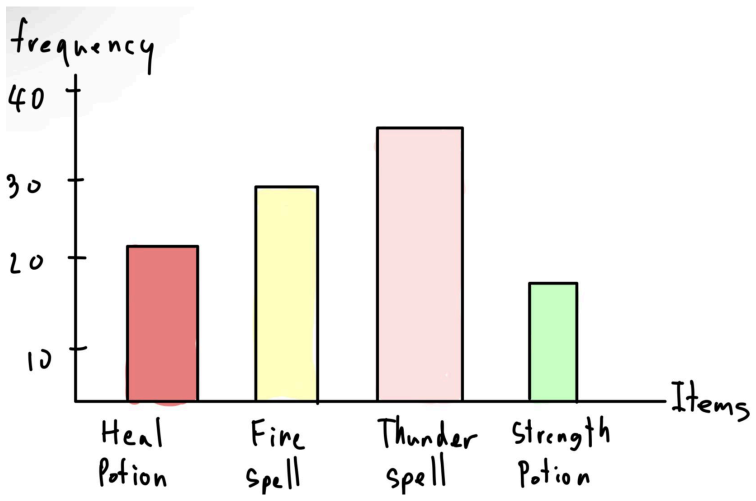
Graph 3



Graph 4



Graph 5



5. Project Timeline

Week	Task
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission / implemented the shooting system, enemy movement, player movement, and player animations.
3 (24 March)	Implemented the level system, boss mechanics, various type of enemy
4 (31 March)	Graphics and Audio Implementation / UI/UX implementations
5 (7 April)	Improve enemy behaviour / testing and debugs
6 (14 April)	Submission week (Draft) / Improve game balance

26 March-2 April	Implement shop/ Implement Magic spells

3 April-9 April	Graphics and Audio Implementation / UI/UX implementations
10 April-16 April	Improve enemy behaviour / testing and debugs / finish all gameplay for a game
17 April-23 April	Balancing a game / Collecting data
24 April-11 May	Data analysis and report

List 50% of the tasks that you expect to complete by 16 April.

- All game feature should be finish

List 75% of the tasks that you expect to complete by 23 April.

- Balancing a game and collect data

List the remaining 25%

- Data analysis and report

6. Document version

Version: 4.0

Date: 31 March 2025

Date	Name	Description of Revision, Feedback, Comments
14/3	Pattapon	The idea is good overall. :) However, section 5 and 6 are missing. Please make sure to add them in the next version.
16/3	Phiranath	Don’t forget to remove the italic format. Missing section 5 and 6. The algorithm section needs a little bit more detail, but good job!
29/3	Phiranath	Good Job! 👍Try not to collect all 50 rows of data by yourself or manually some of them can be collected using time intervals instead to shorten the time taken to play the game.