Slash Mobs!

1. Project Overview

Game concept's going to be about you training as a knight against monsters, getting coins, leveling up and buying new weapons to train better and easier. The game gives players around 3-4 types (may include more) of monsters to fight. Each time a monster is killed/defeated the player level is increased by the indicated amount of exp that type of monster gives. Each mob drops some money after being defeated. The player can use this money to buy new weapons from the shop (I may do a crafting system if I have extra time left to do it)

2. Project Review

This project is inspired by Prof.Chawanat Final computer programming exams. For prof's code, he emphasizes the crafting and resources part, but mine will emphasize on the combat and varieties. (If I have enough time I might do a crafting system just like him maybe 2-3 recipes)

3. Programming Development

3.1 Game Concept

Turn based combat system, mobs attack/skills based on probabilities(using random.choice). Highlight features are varieties of mobs skills and items to buy, I maybe include in a background change for each mob

3.2 Object-Oriented Programming Implementation

Class Player: Store player's data (health, money, levels, etc.)
Attributes: Health, Levels, Money
Methods: Attack, Buy, Killed

Class Monsters(inheritance): Mother class for each mob
Attributes: Health, Money_dropped

Methods: Attack, Dodge, (Their own unique skill, ex. Increase accuracy, evasion, defense, attack)

Example Usage:
Inside each mob do override on def attack or create a new method(skill)

    Class Goblin(Monster): Mob
    Class Orc(Monster): Mob
    Class Vampire(Monster): Mob

Class Background(Draft): for background battle and interface
Attributes: Color
Methods: draw_background, del

Class Configs(Draft): Configure probability for mobs/player's accuracy and many more

Class Items(Draft): Mother class for items
Attributes: Prices, Durability, Accuracy, Damage

Override some stats on items such as prices, durability(might add if able to finish in time)

    Class Sword(Items): weapon
    Class Mace(Items): weapon

Class Shop(Draft): Class for displaying items on sale for players
Class Game(Draft): Put everything to work

3.3 Algorithms Involved

Algorithms:

Rule based logic. If player haven't unlocked a weapon they cannot use it or if player didn't defeat the previous monster they won't be able to try a new one

For fighting mobs I will have an if else check if the player is at the coordinate to fight the mob (Mobs will be placed like a statue on each section of the map)
Maybe I'll have like a self.fight = None, and when you get close to/ on that coordinates self.fight = Goblin then call func() start fight(Goblin.start() or something) and have an if else check whether the mob/player hp is at 0 if so end the fight, add stats to player's attribute then set self.fight = None again

These are just a brief of my game logic, I may change or adjust them to my circumstances.

4. Statistical Data (Prop Stats)

4.1 Data Features

1. Amount of time for player took to defeat each mobs (First encounter) (Total turns)
2. What's the first weapon the player decided to buy
3. What mob the player decided to go against first when starting out
4. What's the first mob that the player died to
5. Amount of enemies defeated before quitting the game

	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)**
Amount of time took to defeat each mobs (first time encounter)	We can see how easy/hard is it to defeat for each monster, so we can balance the game and adjust difficulty for each monster	I can give it out my game to my friends and let them play for a while	Variable turn in class Game	Via summarization statistics (table)
First weapon to buy	Track player's stereotype on what's the weapon they are familiar with/most likely is going to go for the most, so next time if we're making game again we can be sure that this weapon will be used and our efforts of making the weapon will not be wasted	For this one I can just ask my friends if you were to have lists of weapons, which one will you choose to buy first	Variable first_weapon in class Player	Via graph

First monster player decided to go for first	Track player’s stereotype/traits: do they go for the monster that has a weak name like slime? or do they go for the one that sounds cool like vampires? We can see that does monster’s name affect the decision for player	Same method as first weapon to buy	Variable first_monster in class Player	Via graph
First monster player died to	We can know on which state of the game early/mid/late does the player starts to struggle on playing	Same method as the amount of time took to defeat each mobs	Variable first_die in class Player	Via graph
Amount of enemies defeated before quitting	We can know how boring/easy our game is for player to quit at the specific amount of time	Same method as the amount of time took to defeat each mobs	Variable enemies_defeated in class Player	Via graph

4.2 Data Recording Method

I’m going to use CSV file (or SQL if I’m able to learn it in time)

4.3 Data Analysis Report

For this I’m going to use charts, graph or histogram, maybe something else if something is off or if the given example is not sufficient to display my statistics

4.3.1)
(A) Feature: Number of turns taken to kill each monsters type and Number of time a players died to each type of monster (First time only)
(B) Display: Mean of number of turns taken to kill each monster (First time only)

4.3.2)	**Feature name**	**Graph Objective**	**Graph type**	**X-axis**	**Y-axis**
Graph1	First monster player decided to go for	Track player's stereotype/traits: do they go for the monster that has a weak name like slime? or do they go for the one that sounds cool like vampires? We can see that does monster's name affect the decision for player	Pie graph	**Section:** Each monster's name	
Graph2	Amount of enemies defeated before quitting	Track how boring/easy our game is for player to quit at the specific amount of time	Histogram	Range of time played	Number of defeated monsters
Graph3	First monster player died to	Track which state of the game early/mid/late does the player starts to struggle on playing	Bar graph	Monster's name	Number of times player died to that monster type

4. Project Timeline (Project plan)

Week	Task
10 March	Proposal submission / Project initiation
2 April	Study/research on pygame and other modules
9 April	Start and complete all the classes
16 April	Make the game looks playable and add in features
23 April	Finish up the game and add in decorations
11 May	Distribute game/collect data to complete the statistics part

5. Document version

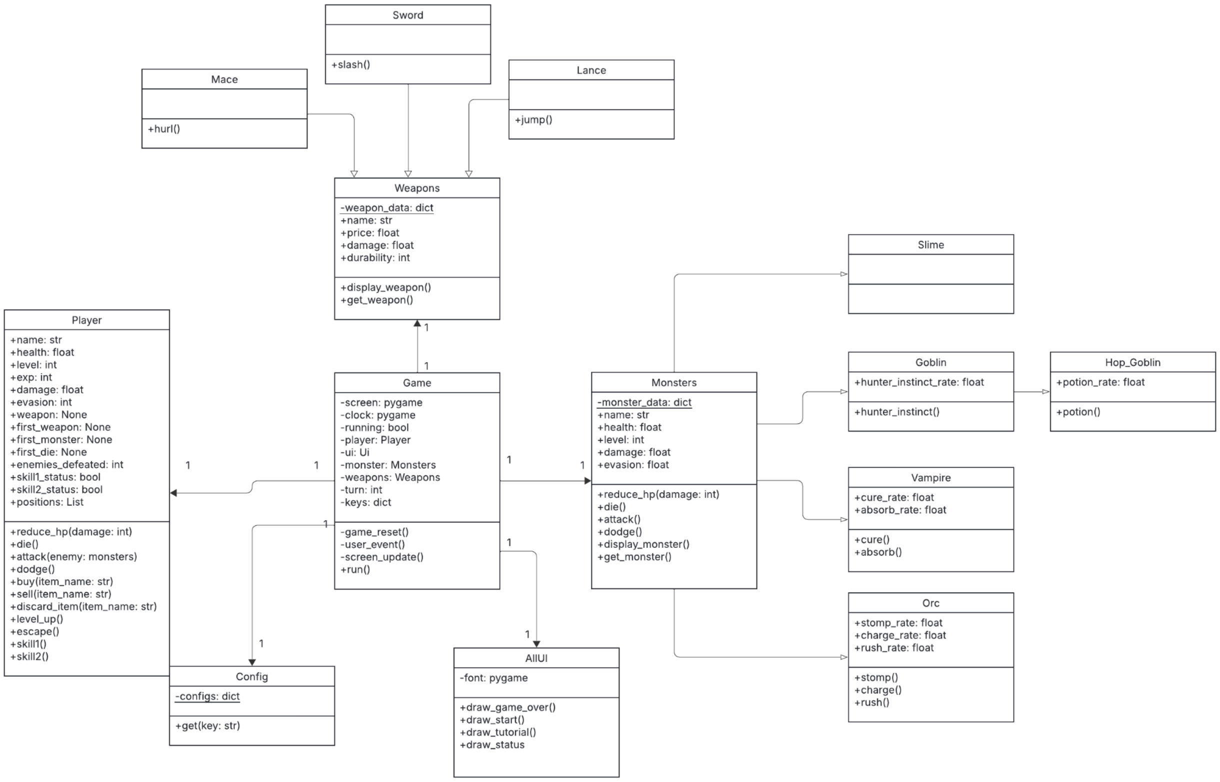
Version: 1.0
Date: 4 March 2025

Date	Name	Description of Revision, Feedback, Comments
15/3	Phiranath	Don’t forget to change the italic fonts to normal fonts. The outline of the Data Analysis Report is incomplete. You need to explain more for example, Total turn player takes to defeat mobs vs The damage of the player current weapons.
16/3	Rattapoom	Please don’t forget to fill the Project Timeline table.
28/3	Phiranath	Overall it is good. Your class diagram relationships are mostly incorrect. Your Monster class is an abstract class but all of the monsters have a composite relationship with the Monster class instead of inheriting or implementing the Monster class. All of the relationships are one-to-one which means that it only has one per one and all of them have only one, so your game would contain only one object for each of the classes. Separated file is good, but don’t forget to add the diagram inside the proposal next time (with the url of the diagram would be really great). ✌️

6. UML (Edited)

Link:

https://lucid.app/lucidchart/46bd8fb2-90ed-4f12-9481-2db04d16d8d1/edit?viewport_loc=-1085%2C-120%2C3837%2C1692%2CHWEp-vi-RSFO&invitationId=inv_48b70261-dffd-4fc1-b561-9abc95d02084


Sword
+slash()
Mace
+hurl()
Lance
+jump()
Weapons
-weapon_data: dict
+name: str
+price: float
+damage: float
+durability: int
+display_weapon()
+get_weapon()
1
1
Player
+name: str
+health: float
+level: int
+exp: int
+damage: float
+evasion: int
+weapon: None
+first_weapon: None
+first_monster: None
+first_die: None
+enemies_defeated: int
+skill1_status: bool
+skill2_status: bool
+positions: List
+reduce_hp(damage: int)
+die()
+attack(enemy: monsters)
+dodge()
+buy(item_name: str)
+sell(item_name: str)
+discard_item(item_name: str)
+level_up()
+escape()
+skill1()
+skill2()
1
1
Game
-screen: pygame
-clock: pygame
-running: bool
-player: Player
-ui: Ui
-monster: Monsters
-weapons: Weapons
-turn: int
-keys: dict
-game_reset()
-user_event()
-screen_update()
+run()
1
1
1
1
1
1
Config
-configs: dict
+get(key: str)
AllUI
-font: pygame
+draw_game_over()
+draw_start()
+draw_tutorial()
+draw_status
Monsters
-monster_data: dict
+name: str
+health: float
+level: int
+damage: float
+evasion: float
+reduce_hp(damage: int)
+die()
+attack()
+dodge()
+display_monster()
+get_monster()
Slime
Goblin
+hunter_instinct_rate: float
+hunter_instinct()
Hop_Goblin
+potion_rate: float
+potion()
Vampire
+cure_rate: float
+absorb_rate: float
+cure()
+absorb()
Orc
+stomp_rate: float
+charge_rate: float
+rush_rate: float
+stomp()
+charge()
+rush()