**The Big Idea:**

For our term project, we want to make a Dungeons & Dragons’ kind of TRPG game before this end of semester. In our game, the main goal for players is exploring this magic world and enjoying the adventure.

First of all, the player will create a unique character by choosing its race, abilities, and the class, for example mages, warriors, and priests. You will form a team with other players or NPCs, and each member can use their own abilities to achieve their goals.

In the adventure, you and your team members will face different situations, and you need to take different actions to correspond them, for example punching an opponent or picking a lock, and the system will use a dice to determine what may happen after you made your choice. Different decisions will cause different denouements. As the game is played, you will change over time and generally increases in capability. You may gain (or sometimes lose) experience, skills and wealth, and may even alter their alignment or gain additional character classes. When you defeat an enemy or accomplish a difficult task, you will acquire XP allows you to advance your level.

In this game, we will try to prove one complete adventure story for our players. We will at least write the whole story plots, and design four different classes in this game, Player class, Event class, Monster Class, and Item class. If we have extra time, besides the main story, we want to add more extra scenes, into this game, and give players different choices to play this game.

**Learning Goals:**

This term project is a good chance for us to learn and practice python. Playing game is our common hobby, so we are really excited to use what we learned to build our first game. For our common expectations, using what we learned in class, and practicing them into a real situation are what we really want to achieve in this project.

For the personal side, learning a new programming structure is really a challenge for me, for example the pygame. But I want to push myself to do more self-learning in programming, and I do believe this project is a good “grindstone” for me. Also, for Ching Chiu, being an independent game developer is his dream, and he also wants to work on game developing after graduated, so this term project is also a good practice for him.

**Implementation Plan**

We have already started using pygame library to build classes and functions that are suitable for our game.

First, we plan to create basic functions that help us build up our menu and our core interface. For example, we have already built Button function, Text function, background with music, and dice functions which support the core event encountering scenes. Both Button functions and Text functions will be used in building our Menu Screen as well as event scenes.

All the actions in our first MVP is mainly driven by clicking buttons. For instance, in a dungeon, the player is expected to have 3 options, **move** [“forward”, “backward”, “leftward”, “rightward”] to explore different routes/rooms, **search** to find treasures/abnormalities in the room/path, and **use** to utilize skills/items. There is definitely more room to expand the interactions between players and the environment/dungeon after the first MVP.

Second, we have to define classes in order to make the game work. For the first MVP, we decide to work on 3 classes, Player class, Event class, Monster class, and Item class.

**Player** class should contain a dictionary of **attributes** {[Strengths], [Dexterity], [Wisdom]}, self.**hp**, self.**exp,** self.**level**, **item** lists = [], and **equipped** dictionary {[weapon], [armory]}. Since we don’t intend to print out the character on screen for our first edition, self.image and self.rect are not needed at the moment. We plan to add more equipment and attributes after MVP.

**Monster** class should be similar to the Player class, except for equipped dictionary. It seems interesting to have monsters actually equipping the items they have but for the first MVP, we will leave it open. The Monster types will be addressed in Monster List and random spawn in the execution section. The probability of dropping the items should also be executed in execution section.

**Item** class should contain the attributes of the item, which means a dictionary of **attributes** {[Strengths], [Dexterity], [Wisdom]}. We will simplify the usage of item to these 3 dimensions. After MVP, we can add different effects, such as attack speed, penetration rate, types of attack. Ideally, the item class should work by just appending the items into player’s item list, and player can choose from its “list” (bag) which items they want to equip or use, and it will be shown in self.equipped.

**Event** class should contain the quest, map type, and possible monsters, but it’s now a little bit vague how we are going to implement it. But for the MVP, we can simply build player, monster, and item first, and use random function and time.tick() function to test if player can actually fight the monster and drop items + exp.

If possible, we would also like to make an **NPC** class, to trigger events, trade items, and skills but it’s not the main concern for MVP.

Thirdly, we have to build our game loop. For MVP, it really depends on how far we will get from the classes. Assuming that we only have Player, Monster, and Item, then the game loop will pretty much like the following: player can click **move**, and the move button will trigger a random loop that random.choice (Monster.List) with a 10% chance of occurrences. If Monster and player collide, then it will launch the battle mode, which involves rolling dices, hitting based on the attribute. The simple algorithm can be monster.hp -= player.attribute[strength] \* (dices)/(18), attack frequency = 20 / player.attribute[Dexiterity], and wisdom is useless in the MVP stage. Every time after winning the battle, monster\_items need to be addressed, and monster.exp will be added to player.exp. If player.exp > 100: player.exp -= 100 and player.level += 1 and player.attribute[i] will add **3**.

The ending point can either be game over or after 15 moves, the player will “walk” out of the dungeon, and screen of Victory will be popped up, stating play again or quit. It can be illustrated by if move == 15:

Victory(), and we have to define the victory screen, which is simply showing 2 buttons, one that starts the game loop, and one that quits the game, and a background.

**Project schedule**

*The first stage*: function and background building

It is expected to be done by this week, and it’s almost finished as we already have buttons, text, background and music, color functions.

*2nd stage*: Building class

Xiang and I are going to split doing the classes. I will be responsible on the formation of characters, menus, monsters, and Xiang will work on events, and create multiple plots and storylines so that players can actually have quests and add some complexities in the game. We expect to spend 1-2 weeks to build these.

*3rd stage*: game loop

Once we have the classes and the plots, it will be easy to add loops based on them. It will basically follow the logics we mentioned in the implementation section, and we expect to have a rough draft by the end of 3-4th week.

*4th stage*: bug testing and feature addition

After having a draft, we can then add more features in our game, for instance, graphics, more complex algorithm and better monster encounter system. We can also add NPCs and more quests in the game, and it’s expected to last till the end of the project.

**Collaboration plan**

Xiang and I are going to split the task up into two sections at the beginning. Xiang is going to work on story plots and event\_lists, and I will be working on the fighting system and other minute functions that may help smoothen the development process. We will be using agile development method, creating MVP ASAP, and try to add more viable features in our game after test-running. Thus, we expect to finish the draft on the 4th week and we can add more features on it. We choose this method because we can better manage our time in the game development. Also, it will be easier for us as developers to debug and test the codes.

After finishing plots and basic elementary coding, we intend to work together on the game loop and how the game functions.

**Risks**

For a game like DND, it is overwhelmingly complicated and complex as it contains battle system, item dropping system, monster encountered system, level-up system, quests system etc. If we try to incorporate every element of it, we may end up having lots of coding to do and yet we may not be able to finish the game development because of our limited coding knowledge and time commitment. We may decide to simplify the game and it may hurt the quality of our game.

Also, designing a good-looking UI is another trepidation which we had. We want to focus more on this game’s contents. Therefore, if the core part of this project takes our more time than we expected, we may not have enough time to learn and design a good UI.

**Course Content**

We believe that more details in how to define **class and dictionary** will be helpful. Also, it will be interesting to know about how codes work when the game needs to connect to the internet or servers (online games), or multi-player games that can be connected via Ethernet, and how can we incorporate those elements in our game going forward.