Rohan Butani

Design Proposal

**Project Description:**

The game that I will be making for my term project is called *Undertown*. In the game the main character explores an infinite dungeon, killing enemies, collecting gold and loot, and becoming stronger until defeated. But there is one catch… I would like to make the game rhythm based such that the main character can only move or attack according to the rhythm of music playing in the dungeon.

**Competitive Analysis:**

This project was inspired by the game *Crypt of the NecroDancer* in which the main character is similarly tasked with exploring a dungeon and can only move according to the beat of music. Similar to *Crypt of the NecroDancer*, my game is also procedurally generated and it also follows the rhythm of music as a main character explores and fights off enemies in a dungeon.

What’s different about my game is that it will be infinite and based more around the concept of survival than of completing some sort of story and secondly, that music will be generated and changed based on the weapon that the main character is using in the game. The other distinction between my game and *Crypt of the NecroDancer* is that my game will follow a room by room map structure as opposed to *Crypt of the NecroDancer* which is not room by room. By this I mean that my game will only display one full room on the screen at a time whereas in *Crypt of the NecroDancer*, the environments displayed are larger than rooms.

**Structural Plan:**

The largest and most general aspect of my structure will start off with class modality. Following this class modality, within each class, I will keep various mode-specific methods and variables. These mode-specific methods and variables will consist of basic functions such as \_\_init\_\_ and \_\_repr\_\_ and they will also consist of functions and variables specific to the objects I have created outside of those classes such as variables relevant to creating different obstacles in each dungeon room or functions to edit or draw those obstacles. I will create several other class objects, besides an obstacles class object, outside of my modal classes such as an object for each different kind of enemy (I am going to do 3 different enemy types and 2 different boss types) which will all be subclassed under an overall enemy class, an object for gold, an object for loot boxes, different objects for different weapons (I plan on having 3 different weapons) all subclassed under one weapons superclass, and objects for different armor (I plan on having 2 different kinds of armor) subclassed under the same armor superclass. Mostly everything else will be contained within my modal classes, particularly my GameMode class. In addition to this modal class I will also have a SplashScreen class, a HelpScreen class, and a TransitionScreen class.

**Algorithmic Plan:**

In this algorithmic plan I will detail how I plan on implementing AI for the enemies in my game.

I plan on making the enemies move in the dx dy of the character as it changes in a timerFired function, but I want to make it such that each enemy has its own movement pattern that it must complete. For enemies that I would like to make easier to fight, I will make their timerFired slower than harder enemies. I will make it such that the movement patterns for enemies are based around each subsequent point they must move towards in order to approach the character. I will have them recalculate the path towards the main character after every iteration of their patterned movement. I will also have it such that if the main character approaches within a certain proximity of the enemies, if the enemies are classified as harder or are bosses, they will break their movement pattern and chase the main character until the main character has gotten far enough away or has killed the enemy. For example, if there is a hard enemy that has calculated the next step it needs to take to approach the main character, and it begins its movement pattern such that it will end on that next step, and the main character happens to come within one block of the enemy, the enemy will detect the character within this one block and break its movement pattern, instead moving straight towards the main character at a speed such that the main character has a chance to escape so that the enemy can forget about the main character and continue moving in its pattern. I would also like to make it such that the enemy cannot detect the main character if there is a block in its path between it and the main character. I will do this by using a similar algorithm as we used in the word search problem we did earlier this semester. I will check every direction around the enemy for the presence of either the main character or a block. If a block is detected, the enemy will break its directional search and begin searching a different direction. Otherwise, if the main character is detected, it will begin its motion that way, searching each direction every time it completes its movement pattern and moves a step forward.

**Timeline Plan:**

Before my MVP next Tuesday, I plan on implementing the following:

-procedurally generated rooms with obstacles, treasure boxes, and enemies

-AI enemies

-health/coin/damage trackers

I would like to have this complete before this Saturday so that I can get a head start on researching how to integrate music into my game. I will then spend some time over Monday and Tuesday and a little bit over Thanksgiving break trying to integrate music into my game so that everything moves according to the beat of music determined by the different weapons the main character is using.

If I cannot finish musical integration over Thanksgiving break, I will put in as much time as I can after returning to try and integrate music before TP3.

**Version Control Plan:**

After every session of work on this project, I plan on uploading my progress to Github.

**Module List:**

-Pygame (Although I may choose not to use it)

**TP2 Update:**

I have not made any sizeable changes to the structure I proposed earlier and have yet to implement what is not already there.