Project Description:

The main part of my project utilizes a computer rendering method called ray casting to generate a 3D game from a 2D board. The board will be randomized each time and the enemies will recursively backtrack to approach you. Enemies will randomly spawn to a set amount of times and in a certain amount of time period. The goal is to reach the end of the level/kill as many enemies as possible (depending on time limitations).

Competitive Analysis:

The game mechanic is very similar to old timey first person shooters. An example of these games would be Doom or Wolfenstein 3D. These games take advantage of a method called ray tracing/casting, which essentially sends out vectors and traces them to then create a 3D image from a 2D map. These games are the earliest first person shooters and utilizes map rendering of 2D blocky maps. My project will be similar to these games in the graphics and in the gameplay, but will have unique sprites and enemies. The design will also be different, and the goal of the game might also differ based upon implementation. While Wolfenstein and Doom carries the player through a map and expects them to finish the full map, I will implement a game that will consist of a player walking around racking up score in a first person sense. Finally, if as a bonus, I will try to implement procedurally generated game boards. If this is successful, then my game will also be more autonomous and varied in the level designs, and have a more complex structure than the old games.

Structural Plan:

The game will have one mainframe that will encapsulate all of the main functions. This is where the lists of enemies, bullets, and graphics will be written. Running the mainframe will run the game itself. The game will then be split into functional groups and objects. Each object will have their own files as each object will have different sprites and animations. Sprites and animations will be stored in folders in the main folder. The important functional groups will also have their own files. For example, backtracking for enemies and drawing the ray casting will have their own files.

Algorithmic Plan:

The trickiest part of the project is the recursive backtracking and the ray casting. Ray casting is initially difficult because of its complexity and novelty. It's a concept that isn't touched upon at all in 112. However, through youtube videos, I can understand the basic concept and implementation of ray tracing, and I will be able to convert that to code (it will take time). The second trickiest part of the project is the enemy movement. Because the player is moving, some sort of backtracking will be called every time the enemy wants to move. This means the movement must be implemented in an efficient way. This will also take time to figure out.

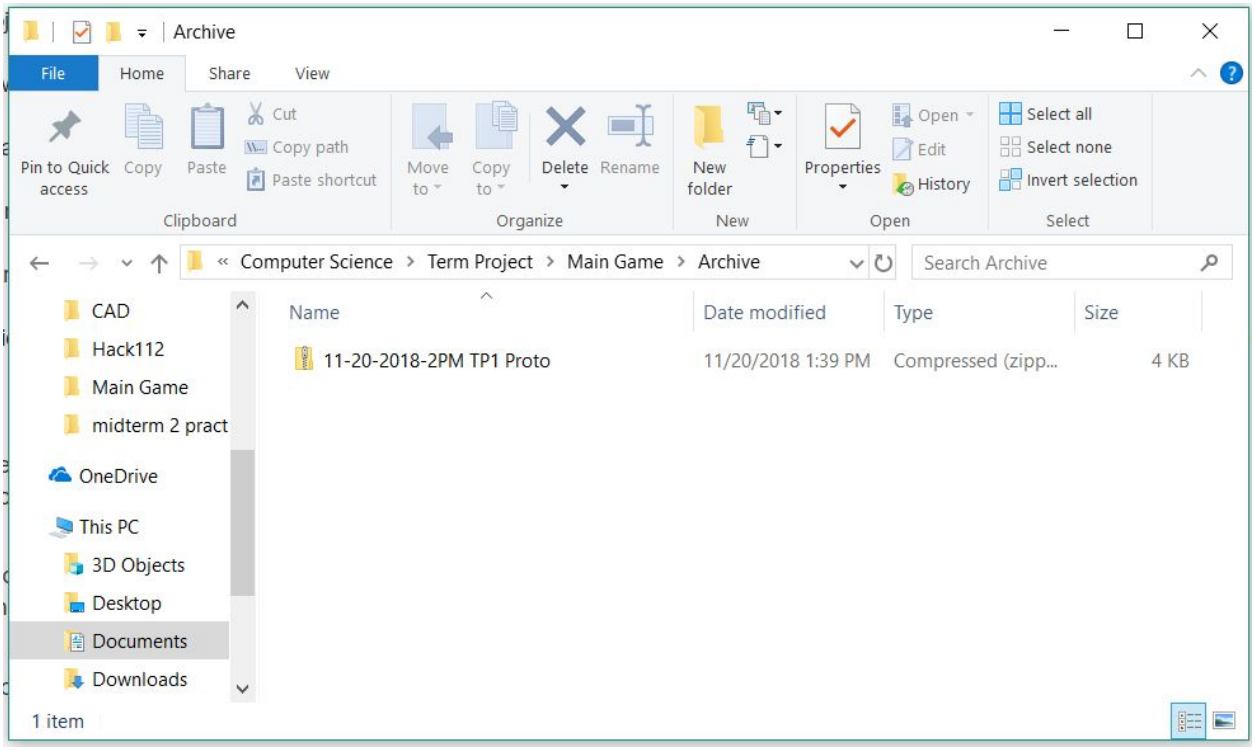
Timeline Plan:

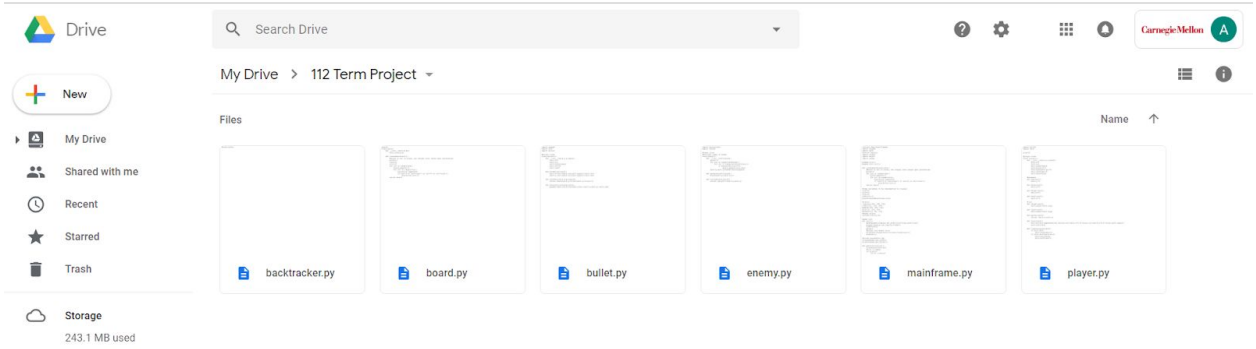
By TP1, the most basic and rudimentary recursive backtracking should be drawn out. This will allow for the enemies to find the player if the player does not move at all. A simple board with simple enemies should also be testable. This means the player can turn, shoot,

destroy enemies, and die. By TP2, a basic ray casting function should be written. This means that the player can see the depth of the board while standing in certain positions on the board. If the function is unable to be implemented or very very very laggy, however, I will instead focus on procedural generation of map to create more complexity to the game. This means that randomized maps are possible and the map will be partitioned appropriately. By TP3, the full game should be complete. This means a fully completed game with simple ray casting and first person shooter mechanics, a game with procedurally and randomly generated maps and enemy variation and complexity, or a game with elements from both ideas.

Version Control Plan:

I will use Google Docs and zip files to effectively manage my versions. This means constant update of google doc documents and having an archive containing all past zip versions(saved every single time work has been done extensively).





Module:

I will use pygame and perhaps numpy for vectors.

TP2 Update:
There are no design changes as of currently. However, the ray tracing runs separately from the main frame. The main frame still proceeds to run the program(the exact same one with all the same variables, with slightly different enemy spawn and move rates for testing) a block tile game while the ray casting proceeds to render the original game in an efficient 3D setting. The ray casting works!
The instructions are the same: left and right arrow to change direction, space to shoot, w and s to move forwards and backwards respectively.