Platform Steve in 2D

# Project Description

The name is Platform Steve in 2D. The name Steve is a nod to Minecraft in which the default skin is called Steve, in that the blocks are square, in the same way the blocks are cubic in Minecraft which is 3D, and also in the way that you can also build and place blocks. I also included the Platform and the ‘2’ in ‘2D’ because it is a platforming game, and also pays homage to platform racing 2, which I based many of the different block types off of. I also included 2D because it is a 2D game. The game will be an arena style platforming game in which you defeat enemies with a sword and a bow, while also having the ability to gain blocks to be placed. There are mobs that try to kill you in a variety of ways, either through melee or through ranged projectiles.

# Competitive Analysis

The two games that my project is similar to are minecraft and platform racing 2. The game is similar to minecraft in that there are mobs that act in different ways, and try to kill the player, while the player is also able to place blocks themselves. It is different to minecraft in the way that the game is two dimensional, and while there are blocks with different functions in minecraft, none are very similar in my game.

The other game that my project resembles is platform racing 2. It resembles that game in the way that blocks are drawn, and the different functions that blocks have, however it differs in that my game is a stationary platforming game and not a side scroller, and also that my game contains mobs while that game does not.

# Structural Plan

The project will be organized in the following ways:

There are two superclasses of objects. These are blocks and entities.

Blocks (typically) do not move, and always collide with the player based on its edge detection. The subclasses of Blocks are the different types of blocks that have different functions when collided with. For example, the ice block prevents friction (or undos, more technically friction) when it is collided with.

Entities, on the other hand, do move. There are three types of entities. There is the player. This is the entity the player controls, and has several abilities unique to it, such as being able to place blocks and have multiple ways of attacking.

The second type are mobs, or enemies. These enemies have pathfinding abilities and use them to do damage to the player through attacks. There are different types of enemies with different path finding and attacking methods.

The third type are projectiles. These are dumb objects that have collision detection when they get in close proximity with any other enemy. They do this by having all other entities check if there is a projectile in close proximity to the entity.

The functions are broken down into MVC, where there are a set of functions that initialize the game and set the game, another which changes the model through AI and user inputs, and the view which portrays the objects in the game. Some of the individual view, and controller functions for the game are contained within the classes themselves as methods. This is so that each function can be different for different types of entities or objects. For example, I have a .drawBlock(app, canvas) method for each block. This can be different for each block, so I only need to call this method when I draw each method, instead of somehow making a complex if statement for the calling whichever function I need to draw without the object.

# Algorithmic Plan

Some of the very hard parts algorithmically are:

Collision detection. Because I am using square blocks and not circles, edge detection is not as simple as comparing the distances between the centers of two objects. Entity detection is much simpler, as it only analyzes the distances between the centers of two entities, and would involve checking the the x and y ranges of its hitbox, then checking if there is any other entity that has a hitbox in that hitbox.

For block collision detection, there are three main aspects. The first is tracing the path of an entity that it can take given a certain dx and dy that it will undergo in the next frame. The collision function then analyzes the collision for each individual box that it will pass through.

The next function in the algorithm is collisionBox. This checks for each point in the entity's movement, what it will impact. It checks the entire bounding box of the entity. If it finds that part of it is inside a solid box, it will then use the function correct displacement which figures out how far inside the box a certain entity is, then readjusts the velocities and placement of the object.

There is also a friction algorithm, that allows it to move while on a surface. This detects if an object is on a surface and prevents it from colliding by setting any downward direction to 0, and slowing the horizontal velocity.

I have not included pathfinding yet, but pathfinding will include a variety of aspects. One of them will be the player - intelligent enemies will try and get closer to the player by walking or jumping nearer it, and if it is close enough it will attack it. It will do this by checking if it has collided or has friction applied in its next instance, and change its behavior corresponding to the conditions it encounters. For example, if an enemy collides with a wall, it will then change its horizontal momentum to the opposite direction.

# Timeline Plan

12/2 - Finishing touches on object detection, implementing projectile detection

12/3 - Implementing advanced pathfinding that can climb on walls, etc.

12/5 - Adding mining and block placing capabilities, attacking, bow functionality, and other advanced ranged mob capabilities.

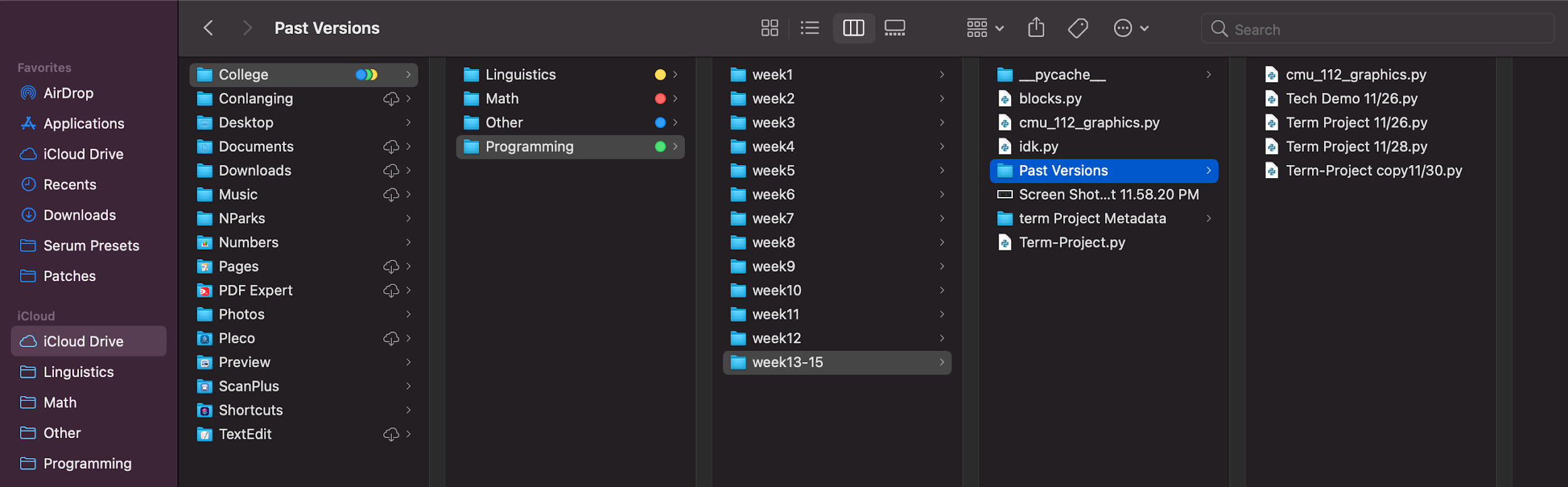
12/6 - Adding additional levels, and possible level creator, as well as improved sprites for blocks and entities

12/7 - Adding boss capable of melee, ranged and block deletion and placement.

12/8 - Adding progression system, and prettier UI.

12/9 - Adding finishing touches

# Version Control Plan

I am using a combination of google drive and also icloud backup. Every day, I save a copy of my file to a past versions folder on my iCloud. Every three days, I save a backup to google drive. I tried using git, but I ended up wiping my computer because of my idiocy :) . So I am just using iCloud as well as google drive to store my data.ta.

# Module List

# TP2 Update

I decided to not use pygame after all. I have also decided to make the game load new areas when you go to the right. Going too far up or down still kills you.