Project Description
Name: Swing 112
Description: This will be a 2D movement/timing game where the character can shoot a rope/hook that can cling to walls/roofs/floating objects and is able to swing from place to place. You must time when to release your swings and when to hook onto the next object in order to navigate the level and avoid obstacles/hitting the ground.

Similar Projects
I was mainly inspired by the swing monkey game on cool math games (Swing Monkey - Play Online at Coolmath Games). In this game, the monkey attaches to the nearest dot with a rope and swings in a circle around it until you let go of the mouse. If your timing is off, you might cling to the wrong circle and swing the wrong way. Once you reach the finish line, you move on to the next level. Because cool math games is mostly for kids, the game is quite simplistic with little to no obstacles/barriers. It is simply based on timing your mouse clicks and releases. For my game, instead of the player being able to attach onto any swing point, there will be a maximum rope length. If the player's distance from swing points is too far, they won't be able to attach on to it. Also, I'm planning to implement different obstacles, such as spikes and lava, in order to make the levels harder. As well, I hope to add in enemies that are moving towards you. You must let go of the rope for a brief second to kill the enemy and then reattach yourself. This makes the margin of error of your timing smaller, making the game more skill-based. Finally, I plan to add environmental features such as wind or different levels of gravity in order to change the physics of the game.

TP0:

Structural Plan

- There's two types of player movement: swinging and non-swinging
    - For swinging, the player will move in a circular motion
    - For non-swinging, the player will move in a parabola

Classes:

- Player: contains lives left, current location, current x and y velocity (dx, dy)
    - Two movement methods (swing and fall)
    - Angle to pivot method to calculate current angle relative to pivot point
- Obstacle: contains location, shape and size
- Pivot class: x and y location
    - Contains canSwing method, which checks if the player is close enough to latch on to this pivot
- Button Class

Model

- List of obstacles
- List of swing points
- app.player

- app.isSwinging: keeps track of if player is swinging or not
- app.livesLeft
- app.ropeLength

View

Different draw methods depending on if its starting menu, game or death menu

- drawPlayer
    - Different animation for swinging vs. nonswinging
- drawRope: draws rope if player is currently swinging
- drawObstacles: draws
- drawBackground
- drawHomescreen
- drawDeathScreen: if app.livesLeft == 0

Controller

- def OnKeyPress:
    - Calls nearestSwingPoint and sets app.isSwing to True if there is a swing point available
    - Makes the player swing

- Def onKeyRelease:
    - Sets isSwinging to False
- Def onMouseRelease
    - Changes app.isSwing to false

- def onStep:
    - Updates obstacles horizontal position since they will move towards the player on each step
        - This will change based on the type of motion that the player is currently in
-

Other functions:

def nearestSwingPoint:

- Finds the nearest terrain circle that the character can attach to

def checkCollision:

- Checks if the player has collided with any obstacles in the obstacle list

def distance:

- This will be used to check which pivot point is closed and to check for collisions

Algorithmic Plan

The most difficult part of the project algorithmically will be the implementation of the physics of the game. The character will need to be able to swing in a circle and a parabola while the obstacles are moving towards them. I will need to do some math/physics calculations to figure out how the x and y coordinates should change on each step.

Another challenging portion will be the scrolling background and level generation. I will need to figure out if I want the obstacles to be randomly generated (a high score system) or if I want the game to be level-based. I will have to think about which data structure is best for storing the current game state/obstacle locations

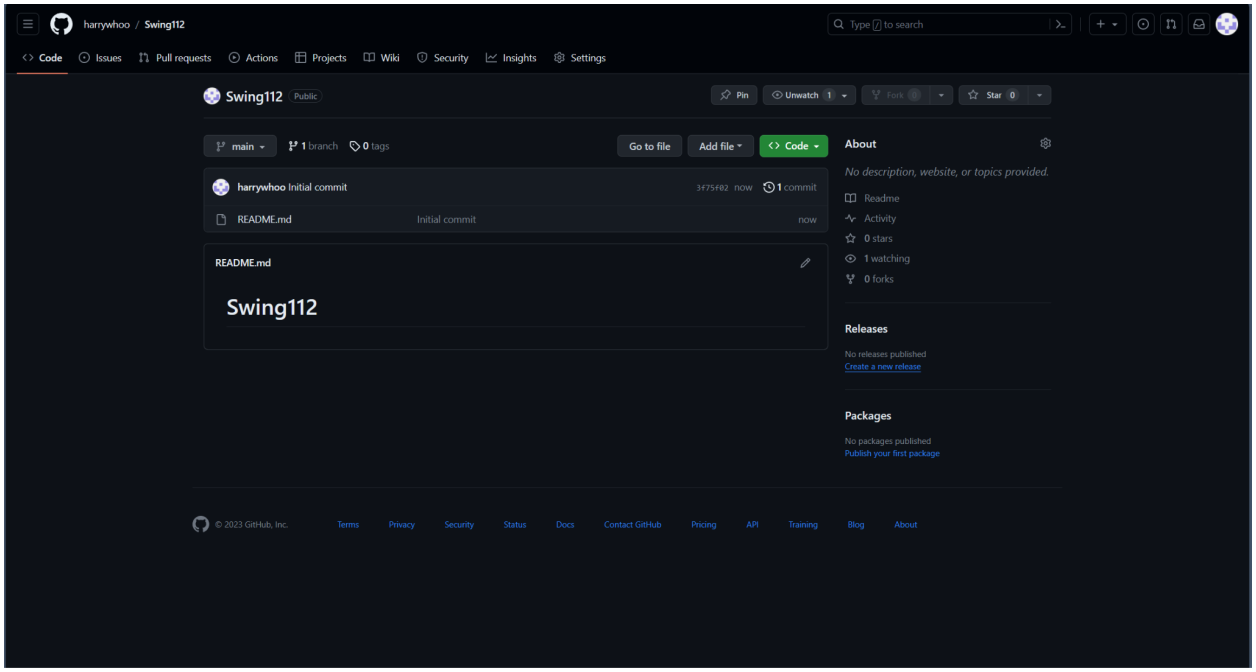
Timeline Plan

- By TP1:
    - Finish basic character movement (swinging, falling through air)
    - Create a health and lives system
    - Figure out the scrolling background/obstacles

- By TP2:
    - Character animations
    - Randomly spawning enemies + combat system
    - Starting screen, death screen, high score screen

- By TP3:
    - Implement harder features such as wind/change in gravity
    - Instead of premade levels, I would like to figure out how to randomly generate obstacles and remembering the game state for checkpoints

Version Control:
Version control will be done using Github. I have already created the repository (shown below).



Modules List:

- N/A for now

TP1:

- No Changes

TP2:

- Completed swing and freefall motion (with player class)
    - Also created some damping so the player doesn't swing forever
- Implemented Spike Class and wall Class with collision checking
- Implemented scrolling background and scrolling obstacles
    - app.scrollX
    - app.backgroundScrollX
    - The game only scrolls when player reaches the right margin
- Implemented pivot point generation
    - Still needs to be refined

- Implemented pivot, wall and spike removal once they're off screen
    - Check if x coordinate < 0, if so → pop from list
- Restructured code to implement setActiveScreen for cleaner code separation
    - Can restart and pause game now
- Added player, wall and spike sprites
    - Player rotates based on swing angle

- Still need to include animations

TP3:

- Reading and writing to high score file
- Added instruction menu
- Improved game UI with buttons and backgrounds
- Implemented random Obstacle generation