Project Proposal Raj Mehta - rajm

Project Name: Active Archer

Project Description: Using your computer webcam, you use two paddles in your hands and imitate the movements of an archer to control an arrow that hits enemies. The goal of the game is to shoot as many enemies as possible without being shot at by the enemies.

Competitive Analysis: There are no games that I have found that are identical to mine. Many Xbox kinect games are similar in that they detect a person's movements and translate that to code. There is a similar mobile game to mine called “spearman warrior.” in which a spearman has to shoot stationary targets, but there is no CV involved in this game. Additionally, a few students last year made games with openCV, so those are quite similar to mine too.

Structural Plan:

Everything is in 1 file, and later I will create another file to store leaderboard values.

Subclasses to Display Modes: Initially there is an opening screen that allows the player to see the instructions, leaderboards, and calibrate their paddle. After calibrating, the game begins.

In the Calibration Screen there is a function within timer fired that allows the user to click on their paddle and it adds the HSV values of their paddle (hue, saturation, value) so that the OpenCV knows what to recognize (explained in more detail below).

The game mode is the largest subclass. Within the subclass we have the appStarted, redraw, and timer fired. There are two classes, enemy and Projectile which allow the creation of enemies and enemy projectiles.

The appStarted just has a list of functions and variables that are initialized

Within the timerFired function:
The openCV converts all video feed from the webcam into frames. Within these frames, the CV tracks all items within the specified range given in the calibration mode and looks for contours. The contours are added to a list if they are big enough and if they are of the right object. The contours have an X and Y position and the timer value constantly updates the distance and angle between these two contours.

Then every 5 seconds, the distance and angle (which correspond to an r and theta value) are used to update the dx and dy value of the arrow. At the 5th second the arrow shoots. And a new arrow initializes when the paddles are close together.

There is also a function called ‘move arrow’ that enacts physics on the arrow. After the arrow is released with the given dx and dy, dy is modified so that gravity acts on it and the arrow rotates by a certain degree of rotation depending on how high it is shot. The calculation of the degree is also in the timer fired function and involves using motion equations (v = u + at).

There are also other functions which run constantly in the timer fired, these are:

placeEnemy: Selects a random location and places an enemy at that position.
shootProjectiles: Randomly shoots projectiles
checkHit: Checks whether or not the arrow has hit an enemy
killPlayer: Checks whether an enemy projectile has hit a person

The last function is redrawAll, which draws all the enemies, arrows, projectiles, etc. so that they are visible on the screen.

Algorithmic Plan, Algorithms I plan on Implementing and the hard algorithms I have:

Currently the algorithms that I have that are difficult are controlling the number of contours and making sure the computer only detects my two paddles and if there are more or less than two paddles, detected, the program should not crash. The way this happens is that it ensures the list of objects detected remains under two. Finding the contours itself is a built in function within openCV, but I had to first convert RGB values into HSV and apply some image processing so that the whole paddle is detected, and that the paddle is detected in different lightings.

The physics algorithms are also quite difficult and require some math to get the gravity right and the rotation. First I had to convert my paddle’s X and Y coordinates to polar coordinates so that I could get a power and angle. Then I had to make sure that the arrow travels at a speed and angle according to those polar coordinates. Then I had to use suvat equations so that the rotation fits with the path of the arrow. I did this by seeing the amount of time it would take for the arrow to reach the highest point of its path, and since the angle is 0 at that point, I can see how much it should rotate until it gets there.

The other hard difficulties I plan on implementing are one that spawns random enemies that can be killed by the arrow. I would generate a random integer within a range using the randint function and if that integer was equal to the midpoint of the range, it would spawn an enemy in a random location. I had to make sure that enemies would not spawn on top of each other by adding their positions to a list everytime they initialized and iterating through the list of enemies before they spawned. The enemy should also be able to be killed. It did this by creating a hitbox around every enemy and continuously checking if the arrow’s coordinates were in any of these hitboxes using a for loop.

These enemies also shoot. This requires the same types of algorithms as above. I need to randomly fire a projectile by an enemy that follows the physics of the game. The same challenges are faced as above.

Some algorithms I will implement later are UI features, such as having a visible health bar and power meter. I will create the power meter by tracking the power and dividing it by the maximum and fill a powerbar accordingly.

I also need to make some sort of dodging mechanism, so the player follows the motion of one of the paddles so that they can move.

I also need to make a leaderboard that can store values of scores and keep them there every time the person plays. They may have to be stored as a set in another file so that they are not removed every time the person restarts the game.

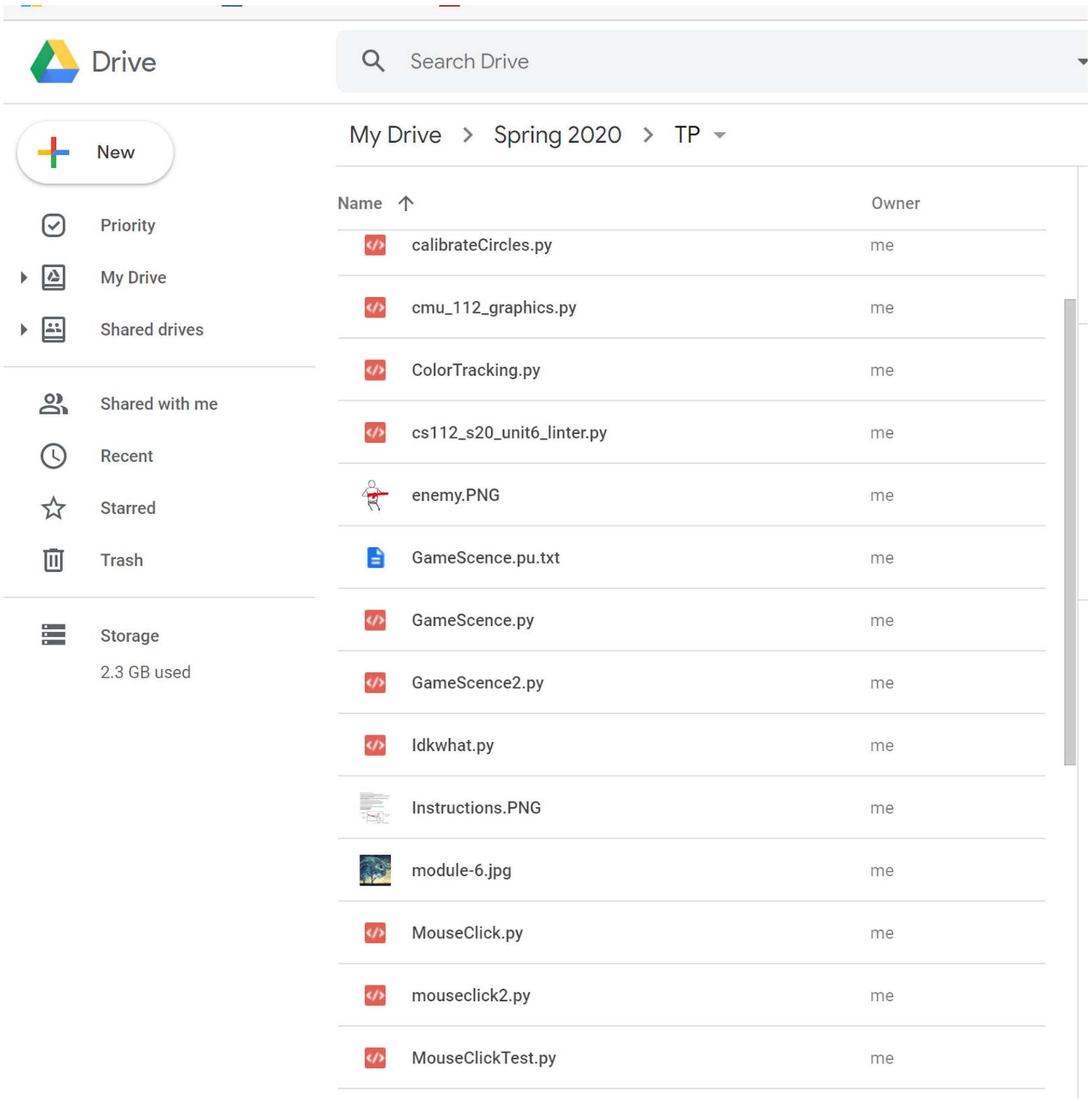
Timeline:

I intend to improve my animations/UI by the end of the weekend (friday 20th April)

I intended to have a dodging mechanism by Thursday 23rd April

I intend to have leaderboards, and additional features such as power ups by the end of the term project.

Backups and Version Control:



My Drive > Spring 2020 > TP

Name	Owner
calibrateCircles.py	me
cmu_112_graphics.py	me
ColorTracking.py	me
cs112_s20_unit6_linter.py	me
enemy.PNG	me
GameScence.pu.txt	me
GameScence.py	me
GameScence2.py	me
Idkwhat.py	me
Instructions.PNG	me
module-6.jpg	me
MouseClick.py	me
mouseclick2.py	me
MouseClickTest.py	me

All my files are backed up, everytime I try something new I create a new file so I have multiple working versions and some that I am trying to implement something new. The names are a little confusing to understand but I keep track of all of them and what I am doing in each.

Modules Used: OpenCV2, numpy, Pillow, tkinter and cmu112 graphics

TP 2 Update:

In terms of design there are no new updates. The changes I have made are that I implemented the features that I said I plan on implementing. My game now has randomly spawning enemies that can shoot, an arrow that kills these enemies, and something that allows the player to dodge enemy projectiles. I also improved my UI. I now have a better UI interface with clickable buttons and colorful backgrounds that make the game look nicer. The health and power bars have also been implemented. I need to work on the game over screen and have some way of storing scores to create a leaderboard.

TP3 Update:

I have created a customizing screen in which players can choose a different player character. I have also added EXP points which is a kind of currency that the player has that allows them to unlock characters.

I have also added a shield. The shield can be obtained by shooting it and grants the player immunity.

There are also two difficulties, normal and hard mode. Hard mode has more enemies and they shoot more often.