Caitlin Kolb

Project Description:

My project is called Soldier Survivor and is a 2.5D catapult game. You are assigned as either an alien or a human and it is your job to either protect (if an alien) or capture (if a human) this new planet that has been discovered by humans. You will build your defense using blocks (made through raycasting) and hide soldiers on your side. Once each time has finished hiding their soldiers, each side will have a catapult to launch fireballs to the other side.The first side to kill all the soldiers wins.

Competitive Analysis:

Soldier Survivor will be somewhat similar to classic tank games such as Scorched Earth. It will similarly have two sides, each launching at each other. However, my project will vary from these in that it will use ray casting to give a 3D look so that it has a first person view, while these games are made in 2D looking from the side.

There are also similar 3D tank games such as Tank Hero where there is a similar object to kill the other team. My game will be similar in that there will be barricade type objects to hide behind in order to make it harder to hit the other team. However, my game will differ in that you will be able to make your own defense instead of having levels with preset different defenses. My game will also add the aspect of the planet story in order to make it more engaging.

Structural Plan:

My game will have 4 modes: start screen mode, instructions mode, story mode, and set mode. Each mode will be in the main function. I will have separate files for the raycasting and projectile as they will likely have lots of code involved. I will also have a separate file for moving around and stars as this code is repeated throughout the game.

My Modal App:

Functions:

App started: sets the modes

- Start screen mode
    - Functions:
        - appStarted
            - Set necessary variables
        - mousePressed
            - Check if pressed button
        - mouseMoved
            - If hover over button, highlight it
        - redrawAll
            - Class stars and draws button
- Instructions mode
    - Functions:
        - appStarted
            - Sets background color

        - mousePressed
            - Checks if pressed button
        - mouseMoved
            - If hover over button, highlight it
        - redrawAll
            - Calls stars and draws button
- Story Mode:
    - Functions:
        - appStarted
            - Sets necessary variables
        - timerFired
            - Changes text when necessary
        - mousePressed
            - Checks if pressed button
        - redrawAll
            - Calls stars, draws button and story
- Set Mode:
    - Functions:
        - appStarted
            - Sets necessary variables
        - mousePressed
            - Places block
        - keyPressed
            - Calls move file
        - redrawAll
            - Calls raycasting
            - Calls projectile
            - Calls collision detection

Algorithmic Plan:

The most difficult parts of my project are creating a 2.5D world as well as the projectile launching/ interacting with the blocks.

2.5D world:

- Implement raycasting in order to create the 2.5D effect
- Use keyPressed in order to be able to turn.
    - Use vector addition and check if can rotate/ move by checking if there is a zero in that place on the map
- Use mousePressed in order to place the blocks
    - See if there is already a block there, and if there is place it on top
    - Use vector addition with current position and direction facing and use the event.x and event.y to place blocks in the correct place

Projectile launch:

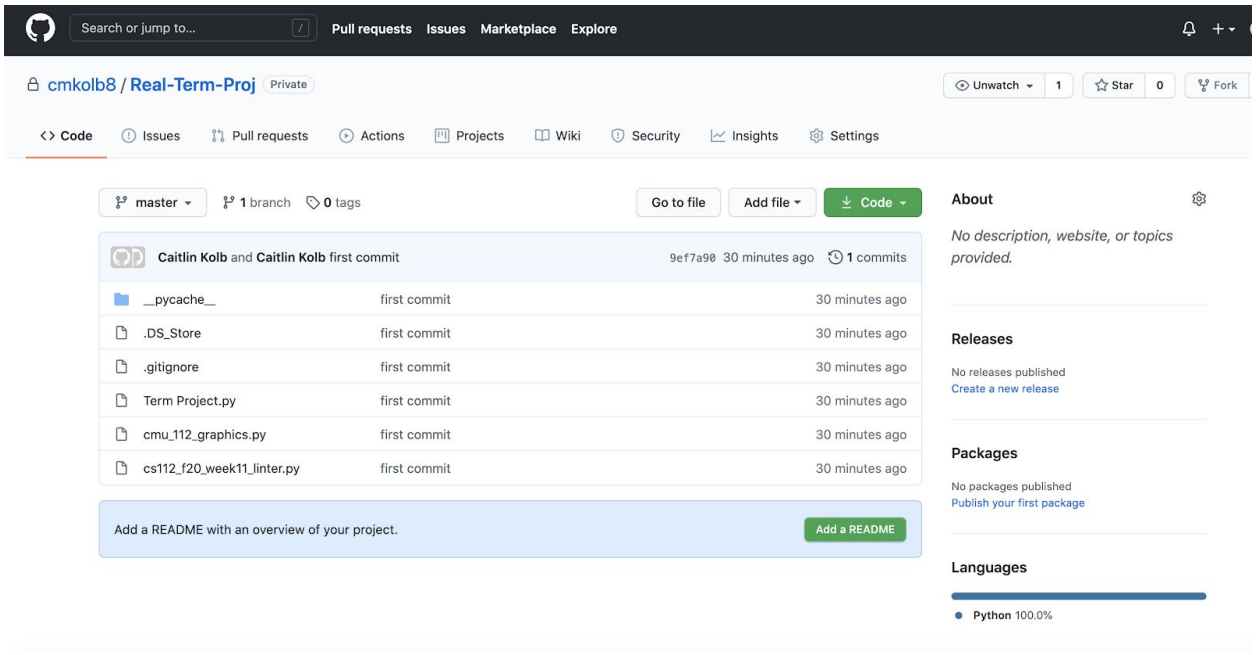
- Use velocity, position, acceleration, and distance in order to calculate where the projectile should land based on how far back the player pulled the lever
- Then use distance formula to calculate whether the projectile will hit a block
- If it hits, insert a fire image and delete the block (by putting a zero in that spot on the map) so it looks like it blew up
- Then put an image of rubble where the block used to be in order to show how it was destroyed

Timeline Plan:

Nov 30: Work on bugs in raycasting and begin thinking about/ implementing enemy defense.	Dec 1: Finish raycasting and block placement and work on enemy defense	Dec 2: Finish enemy defense generation. Figure out and implement the catapult and the shifting from the set mode to game mode.	Dec 3: Work on the physics of the projectile/ clashing with the blocks, as well as the opponents projectile	Dec 4: Work on/ finish the physics of the projectile/ clashing with the blocks. Begin win/ lose detection.
Dec 5:	Dec 6:	Dec 7:	Dec 8:	Dec 9:
Finish win/ lose detection.	Floor raycasting?. Improve the UI of the collision	Improve the opponent	Improve the opponent	Finish up any bugs/ fix UI.

Version Control Plan:

I uploaded my code to gitHub and will push them to the cloud whenever I have made major changes.



Module List:
Pygame (will likely use for music)

TP2 Update:
I changed the soldiers to green blocks which are trying to be protected.

TP3 Update:
Added a top view of the board (press ‘t’ to see and ‘r’ to exist)
Added sound effects and music throughout the game
Added an easy and hard mode
Made the game easier to win as it was very hard before