Ishraaq Shams
15-112 - Kosbie
Project Proposal

BeatHazard.py

Project Description:
Beat Hazard is a two-dimensional space-shooting game that analyzes a song, and based on its intensity, it spawns enemy ships and asteroids or different sizes as obstacles. The objective of the game is to stay alive for the duration of the song.

Competitive analysis:
To learn the basics of PyGame for the tech demo, I have analyzed the code for a simpler arcade game: Space Invaders. Although they are both space-shooting games where the player shoots at the enemy obstacles in order to stay alive, Beat Hazard takes in an audio input and spawns its enemies in a different manner. Both the Space Invaders game I was examining and the Beat Hazard game that I am planning to make uses pyGame to a significant extent in order to move each object around, for example the player, the obstacles, and the bullets. PyGame also allows both to play audio clips and to incorporate sound effects for bullet fired and bullet collision. In terms of collisions, PyGame allows both games to detect collisions and process images more efficiently than a module like tkinter. Additionally, both of the games have similar controls for motion using the arrow keys.

Beat Hazard has more features than Space Invader. The most obvious one is that Beat Hazard has music playing in the background that generates the obstacles based on song intensity. In the game play, the player in Beat Hazard is not limited to moving left and right like the player in Space Invader, but the player can move everywhere in the frame. The user can choose what song he/she wishes to play and also can select the difficulty of the game. There are more features like a health bar for the player and power ups which can be attained through shooting the enemies. Lastly, Beat Hazard also uses mouse controls to aim at the enemies.

Structural Plan:

- Object Oriented Programming for the Game Objects
    - Classes include: Player, Enemy, EnemyShip, Blast
    - Subclasses include: smallShip, mediumShip, largeShip, Asteroid
    - Contains the images and the bounds for each object
    - Contains information of the health left and damage done
- Game loop stored in a class:
    - Runs the changes of position of the objects of the game.
    - Contains the wav files for collisions and bullets fired
- Each song is a separate class that stores information of the song, for example: the volume of the song, percentage of time played and time left
-

Algorithmic Plan:

- Levels of Difficulty (Easy, Medium, Hard):
    - Change the frequency of enemies being spawned and the number of enemies being spawned at a time
    - Easy will have the lowest frequency and Hard will have the highest
- Player and Enemy rotation
    - Player Rotation - depends on cursor
        - In while loop for game, it keeps track of the cursor position and finds the angle created by the line between the center player position and the cursor position using math.atan(change in y / change in x)
        - Then, rotate the image based on the angle
    - Enemy rotation - depends on the player position
        - In while loop it does the same thing as player rotation and calculates the angle the same way as player rotation
- Shooting the bullets
    - Create a list of the total bullets on present in the screen
    - It is much harder compared to Space Invader now that bullets can be fired at an angle
    - Keeps track of the initial player center position and the mouse clicked position.
    - Find the unit vector between those two points and multiply it by the velocity of the bullet
    - Then in drawAll in the while loop, update the position of the bullet by adding the new x and y changes found above
    - When the bullet is off of the screen, remove it from the list
- Power Ups
    - When the player collides with a power up, power up immediately takes effect
    - The power ups are:
        - Increase Player Speed
        - Increase Player Health
        - Increase player bullet damage
        - Destroy all enemies on screen
- Enemy Position and tendencies
    - Patterns of entry for small and medium ships
        - These ships can come in clusters back to back around the same point. Generate a random point on the screen boundary and have ships be spawned equally spaced one after the other from that one point

Timeline Plan:

- 11/20 - By TP1 - Create Player image - rotate with cursor and move around. After TP1 - Implement shooting blasts in the direction of the cursor
- 11/21 - 11/22 - Finish up shooting. Create Asteroid,
- 11/23 - Enemy ships and and enemy shooting, Enemy spawning and motion , health bar,
- 11/24 - 25 - implement Easy Medium and Hard Modes , create Splash screen , Win Screen, Game over Screen, Pause screen, and Help screen, Start Power Ups and its effects
- 11/26 - TP2 - start looking into pyAudio - implement scoring, detailed instructions and song list screen
- 11/27 - Create a list of songs to choose from before playing the game. Set the duration of the game to the song length
- 11/28 - 11/30 - Analyze the song and its intensity, match the enemy entrance with the beat of the song
- 12/1 - TP3 Deadline - Polish the game / audio analysis

Version Control - github repository https://github.com/ishraaqsshams/15-112-Term-Project

Module List:
PyGame - mixer
After MVP - PyAudio or aubio