Sumama Haque
Final Document for APCS Project
6/8/23

Period 1
Group member name: Sumama Haque
Group name: verybiggiantboi

Project title: Sumama Haque's Supercar Rampage

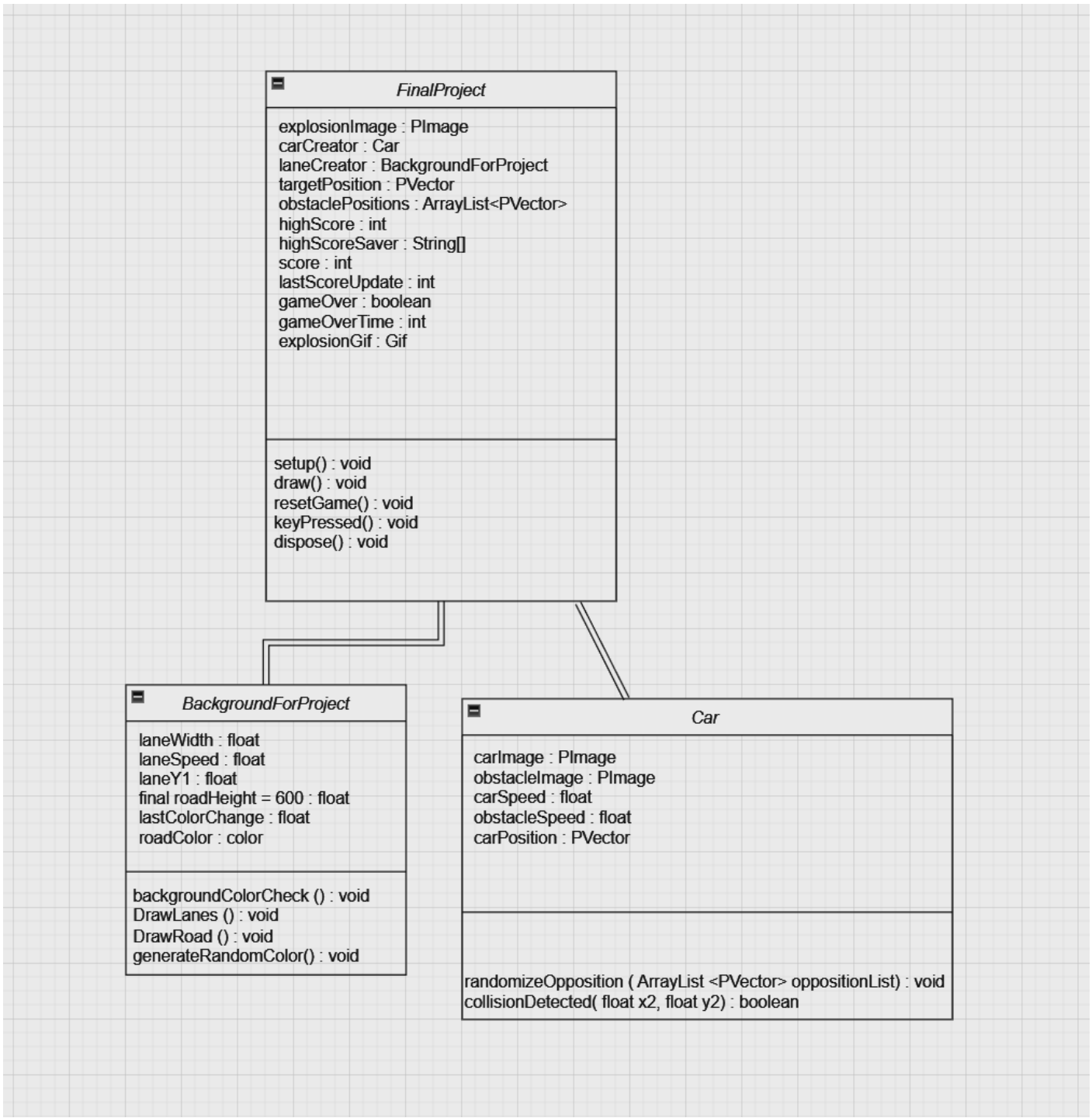
Brief Project description:
My project is a seemingly simple car game. Specifically I modeled it on the endless car games where you have to avoid the incoming cars. It is modeled on the countless car games on the App Store and Google Play for children, and making this game brought back countless childhood memories of me playing these games on my iPad for hours.
In this game, you are driving a yellow Honda Civic VTEC in a city called Endless Utopia. However, the city is endless, and you are unfortunately trapped forever, but you are spawned immediately into the game and you want to get out anyway. You find yourself on the wrong side of the road, but you must avoid all the cars, (to make it more nerve wracking, they're all the same model – the Mercedes - Benz SLR McLaren), otherwise you restart.

Detailed Functionality Description -

- There are three Classes, each with their different purpose, as well as 3 images, one of the Obstacle Car, one of the PlayerCar, and one gif animation of the explosion.
- The First Class is called FinalProject, which is the main class.
    - This main class is a sort of a placeholder class for the functionality of the other classes, BackgroundForProject class, and Car class. To reserve values and to manipulate variables using procedures for these classes (which is described further below) there are 2 variables that contain a reference to these classes – laneCreator (BackgroundForProject class) and carCreator (Car class).
    - However, the main class has many of its own special functionalities. In Setup() , images are loaded and if it's a car image, it is loaded onto a variable into the carCollector reference of Car class. The gif image is stored in a variable called explosionGif, which is a reference to the Gif class.
    - The Draw class is where all the functionality from all the classes come together. The Draw method has
        - Methods from the BackgroundForProject Class that creates the road, changes it every 15 seconds, and creates the yellow rectangles that are like the yellow road dividers.
        - The Draw method updates the speed (laneSpeed) of this drawn yellow rectangle, which creates an illusion of movement.

        - It also calls functions from the Car class, to randomize opponents and to call for collision detection
        - However, the Draw method stores the arraylist of the Opponent Cars, and draws the images of the cars.
        - The Draw method also uses an interpolation technique, to smoothen the movement of the PlayerCar.
        - It displays the current score (1000 for every car passed, 10 for every millisecond) and updates the current high score. The last high score is saved in a highscore.txt file, and is read from the setup(). If score exceeds highScore, highScore becomes score and is saved to the txt file to be read the next time you play.
        - It also contains a gameOver boolean, which turns true if collision detected, and shows an explosion animation at the site of crash, as well as a message saying you lose and press shift to reset.
    - Reset method resets the game, setting original speeds, and resets the score to 0.
    - keyPressed method
        - **Pressing left moves PlayerCar left by the amount dictated by current carSpeed within the boundaries of the road.**
        - **Pressing right moves PlayerCar right with the same conditions as left.**
        - **Pressing up increases laneSpeed, carSpeed (sideways car speed) and obstacleSpeed (deliberately set as the same speed as laneSpeed) by 5.**
        - **Pressing down decreases these speeds by 5.**
        - **Shift Key resets game**
    - Dispose function disposes of the explosion gif after use.
- The second class is called BackgroundforProject and it handles background functionality.
    - It draws the road, and draws the yellow lane rectangles,
    - Through the function backgroundColorCheck (), it checks whether or not 15 seconds have passed, and if true, it changes a variable called roadColor, which then changes background color.
    - There is a generateRandomColor() function
    - There is a constructor, which controls laneWidth, speed and size.
- The third class is called Car, which handles car images, as well as sending cars.
    - It has PImage variables to store the cars images, and stores car speed and obstacle speed. It has a PVector variable to store PlayerCar's vector. The movement functionality is handled in draw().
    - It has a randomizeOpponent() procedure that sends a random car at a rate of 0.005. If the cars overlap, it will look weird and will be impossible to overtake, so it will break if their random PVector value overlaps. If the test passes, it adds it to the PVector arraylist of obstacleCars in the main class.
    - It also handles a collision detected procedure to be called in the main class, which is used in draw() as a boolean to set gameOver boolean variable to true.

UML Diagram:



How it works:

- Press play on processing, and you are spawned into the game.
- Move left with the left key, and right with the right key. To make the car go faster, use the up key, and if you think that the car is going too fast, use the down key.
- Avoid the coming cars. If you don’t, you will lose and your score will be reset. If you lose and want to play again, press the shift key. If not, just close the window.
- Every opponent car passed is 1000 points, and every second is 100 points.
- Every 15 seconds, expect a background change.