Team 09

# PyRace

Tommy Kaplan, Dylan Klasing, Tristen Shields

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Operation Catapult 2021

## Introduction:

Our project assignment was to learn how to code in a programming language called Python, and then to create a game using Python code. Python is a programming language that is growing in popularity because it allows for easy access to others' code by importing packages of code called "libraries", which can make coding substantially easier by providing built-in functions. The game was made using the Python library "Pygame", which is designed to allow programmers to easily make games in Python code.

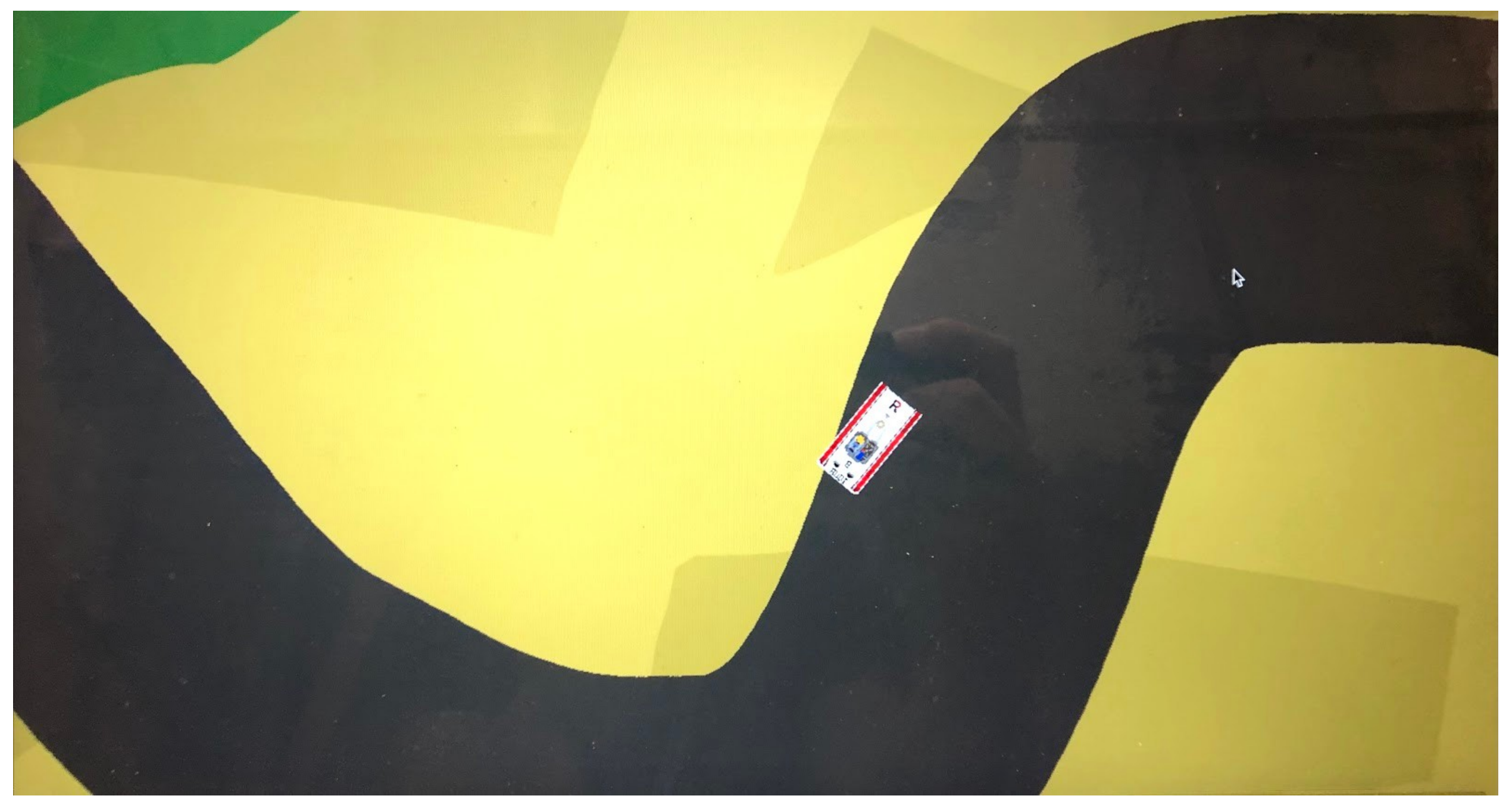
## Problem Statement:

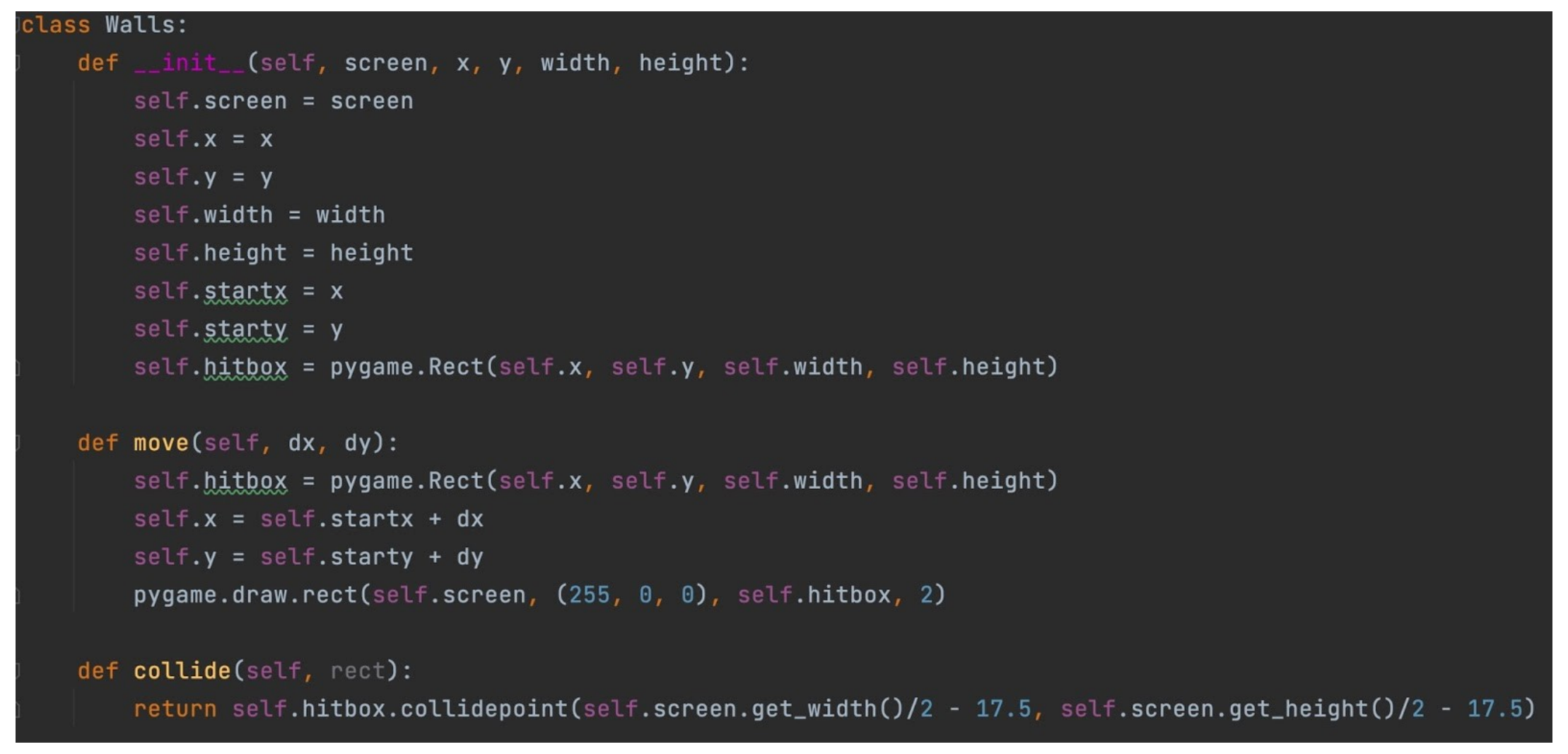
For our project, we set out to build a two-dimensional racing game in Python code, hence the name "PyRace". It will work by allowing the user to accelerate and decelerate a race car by pushing the 'W' and 'S' keys, which won't make the car itself move, but will make the track around the car move. It will also allow the player to turn by using the 'A' and 'D' keys, which will rotate the image of the car and change its direction of motion relative to the track.

## Background Information:

Projects written in Python coding are often organized in "classes", which often represent certain elements of the game. These classes are then broken down into "functions", which allow the code to perform certain organized tasks. Different classes will have different variables used to perform different functions within the class. All of this is executed by one "main" function at the end of the code, which goes and executes all of the functions in those respective classes in the intended order.

PyRace Final Logo

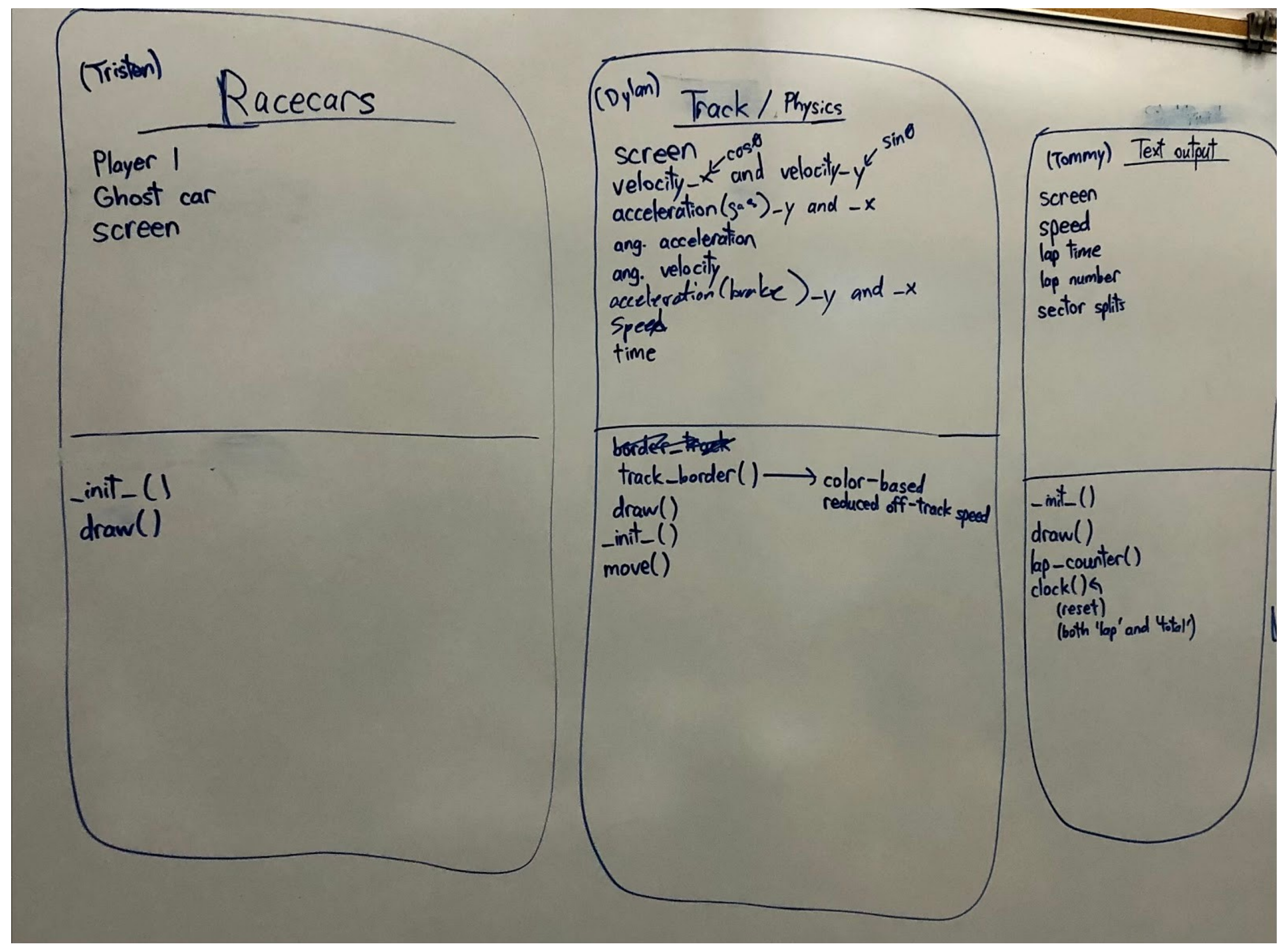

Track During Development



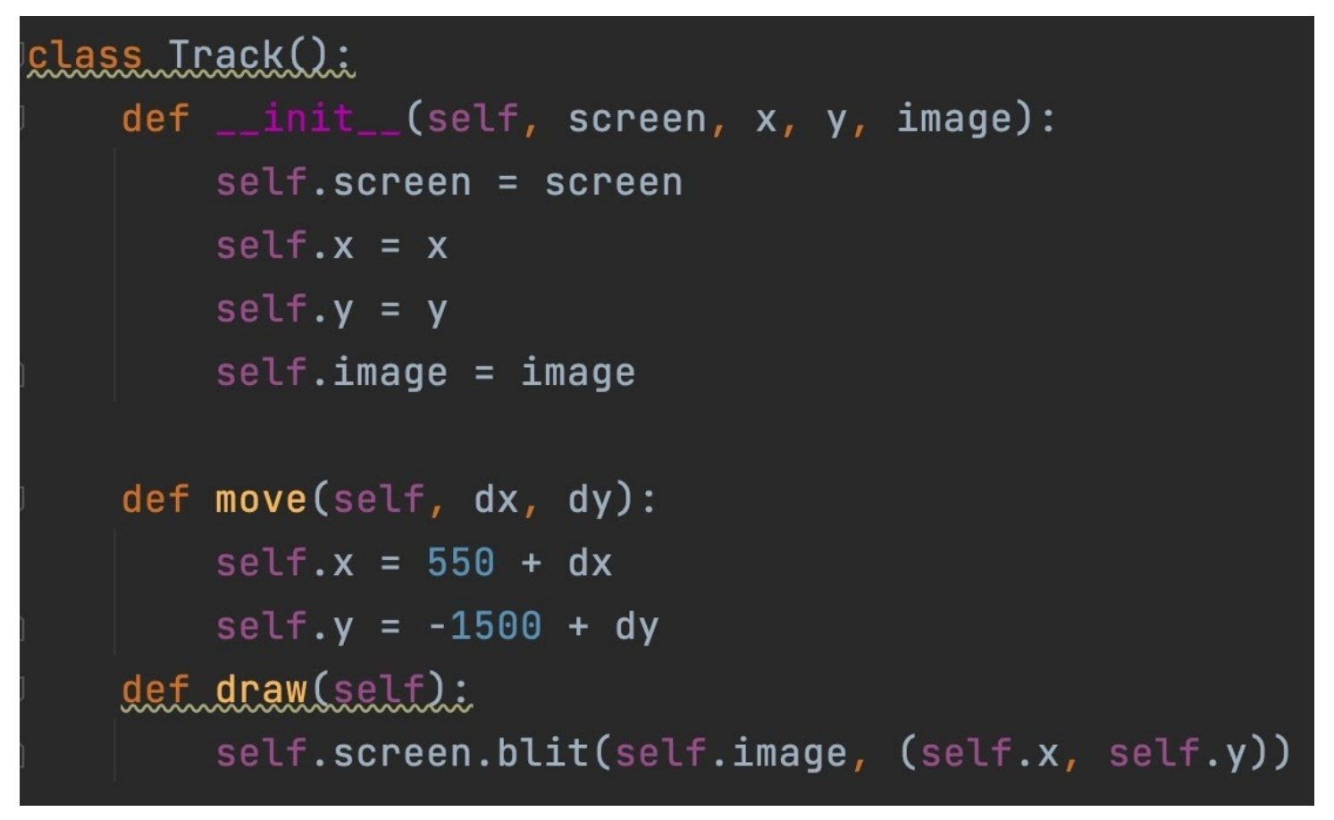
```python
class Walls:
    def __init__(self, screen, x, y, width, height):
        self.screen = screen
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.startx = x
        self.starty = y
        self.hitbox = pygame.Rect(self.x, self.y, self.width, self.height)

    def move(self, dx, dy):
        self.hitbox = pygame.Rect(self.x, self.y, self.width, self.height)
        self.x = self.startx + dx
        self.y = self.starty + dy
        pygame.draw.rect(self.screen, (255, 0, 0), self.hitbox, 2)

    def collide(self, rect):
        return self.hitbox.collidepoint(self.screen.get_width()/2 - 17.5, self.screen.get_height()/2 - 17.5)
```

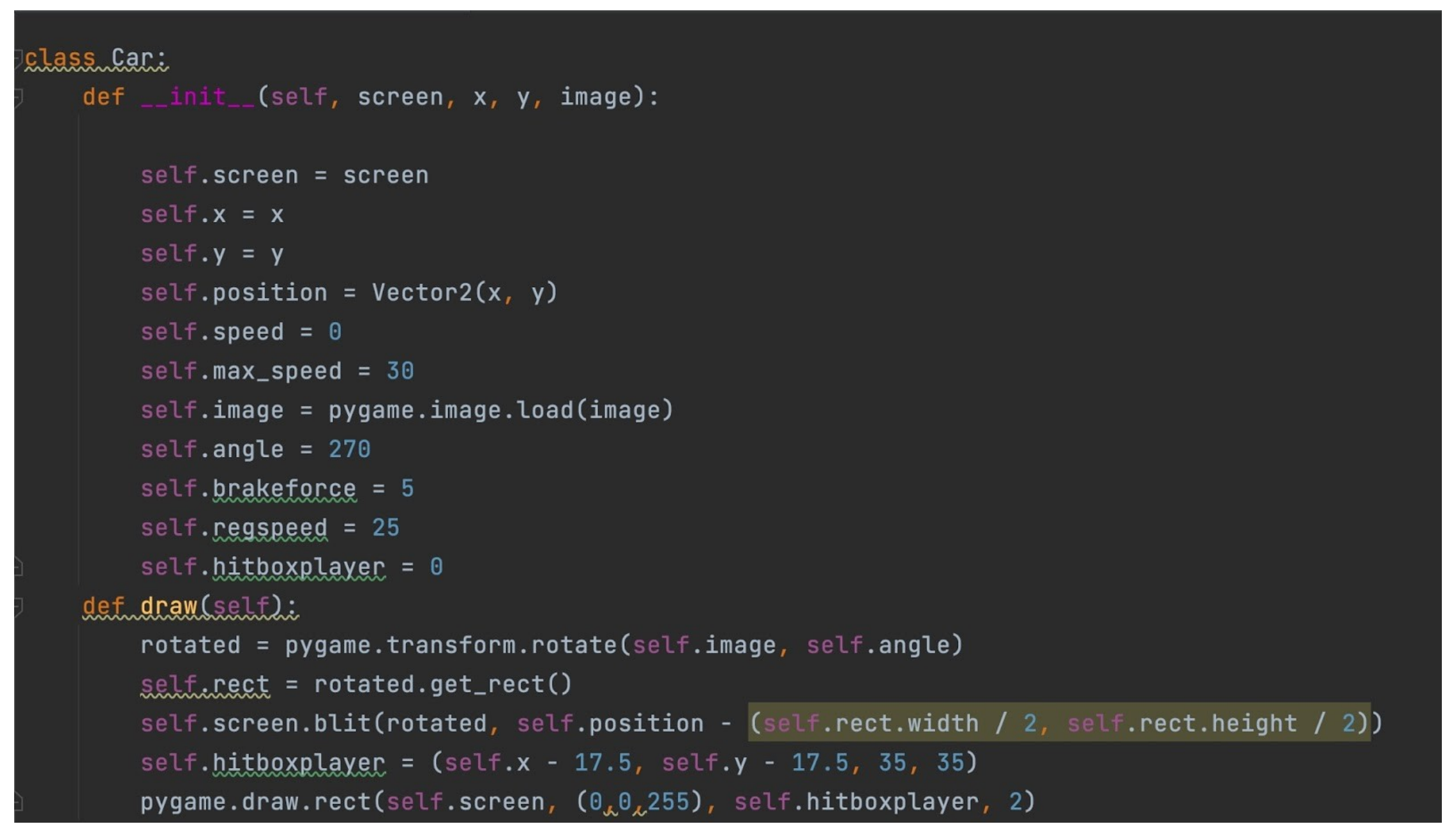


First steps of development (planning the classes of code)



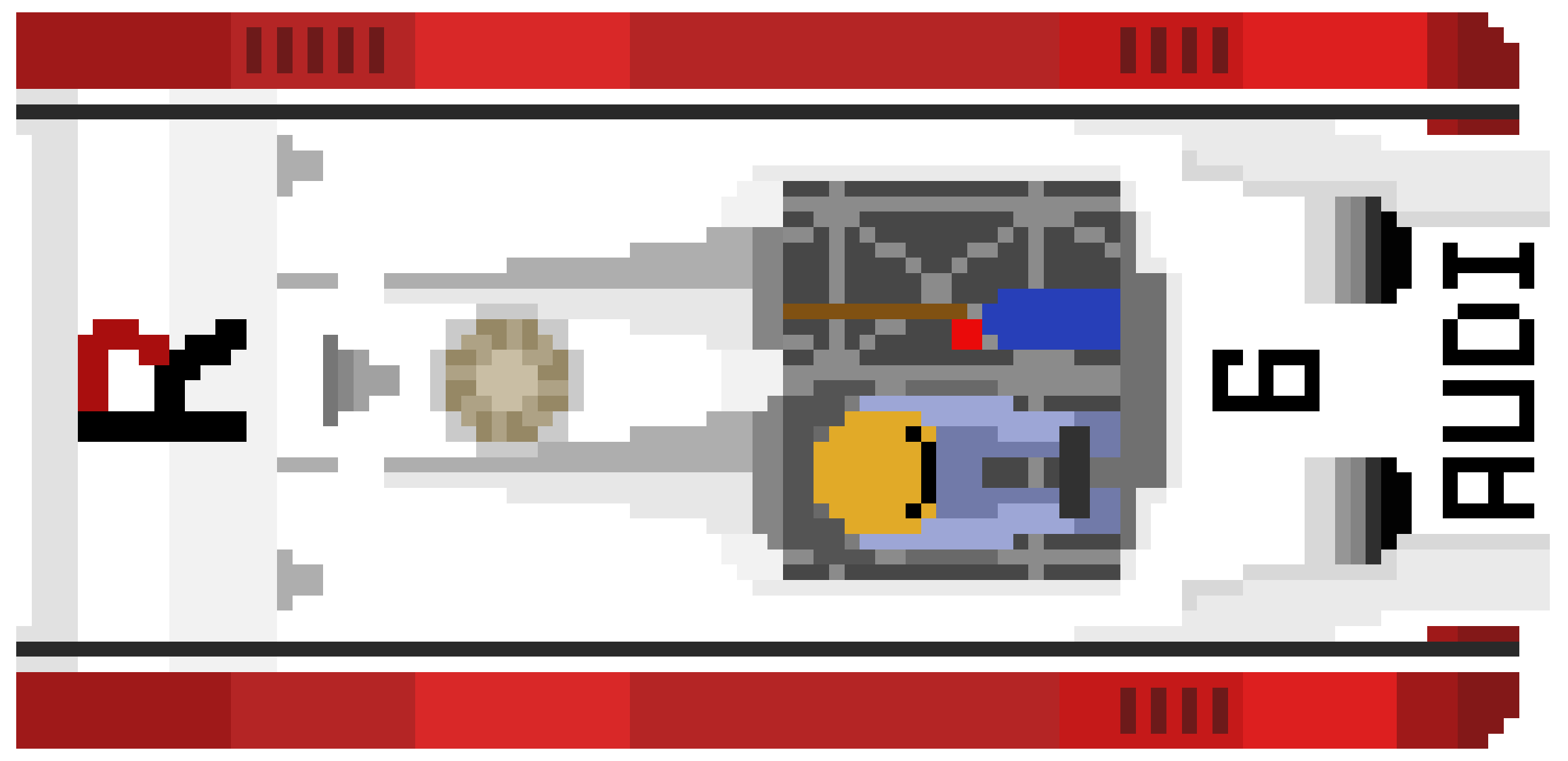
```python
class Track():
    def __init__(self, screen, x, y, image):
        self.screen = screen
        self.x = x
        self.y = y
        self.image = image

    def move(self, dx, dy):
        self.x = 550 + dx
        self.y = -1500 + dy
    def draw(self):
        self.screen.blit(self.image, (self.x, self.y))
```



```python
class Car:
    def __init__(self, screen, x, y, image):

        self.screen = screen
        self.x = x
        self.y = y
        self.position = Vector2(x, y)
        self.speed = 0
        self.max_speed = 30
        self.image = pygame.image.load(image)
        self.angle = 270
        self.brakeforce = 5
        self.regspeed = 25
        self.hitboxplayer = 0
    def draw(self):
        rotated = pygame.transform.rotate(self.image, self.angle)
        self.rect = rotated.get_rect()
        self.screen.blit(rotated, self.position - (self.rect.width / 2, self.rect.height / 2))
        self.hitboxplayer = (self.x - 17.5, self.y - 17.5, 35, 35)
        pygame.draw.rect(self.screen, (0,0,255), self.hitboxplayer, 2)
```



Final Car Model

## The Design Process:

First, we organized the game into three main classes: 'Car', 'Track/Physics', and 'Text output'. 'Car' put an image of a custom race car on the screen and located on the track, 'Track/Physics' contained the track and the systems of motion that allow the track to move along with the image of the car. 'Text output' simply displays the speed, lap time, and rankings. These weren't our final classes, but they allowed us to organize the project early on. Once the layouts of these classes were written in code, we worked on drawing a track, and implementing our physics (motion) system on that track. At first we had a curvy track with a desert background, but after running into issues with pixels being out of range due to its complex shape, we drew another track that ran more like a maze in order to make coding colliders easier. Once we could move the car along the track, we added the text output code, background music, and track borders ("colliders").

## Results:

While the game looks different than we originally intended, due to issues regarding setting up track borders for complex shapes and limiting pixel ranges, we succeeded in making a two-dimensional racing game.

## Reflection:

Over the course of our design process, we had to try six different methods of instituting colliders. We spent days figuring out how to set up colliders without running into technical issues with our game. In addition, we had to change the  entire map design before we could implement the correct method. In conclusion, this project required us to be very adaptable and very persistent.