Final Project for semester 2 python

Name: Kshitij Chandrakar
Course: B.Tech CSE (Hons)
Semester: 2
SAP: 500124827
Enrollment Number: 2142231661

Topic: Video Game Using Computer Vision

About the project:

As the name suggest, its a platform based runner game, however the controls are based on the movement of the user as captured by the camera.
it utilizes various technologies like: - Movenet - Tensorflow - Computer Vision

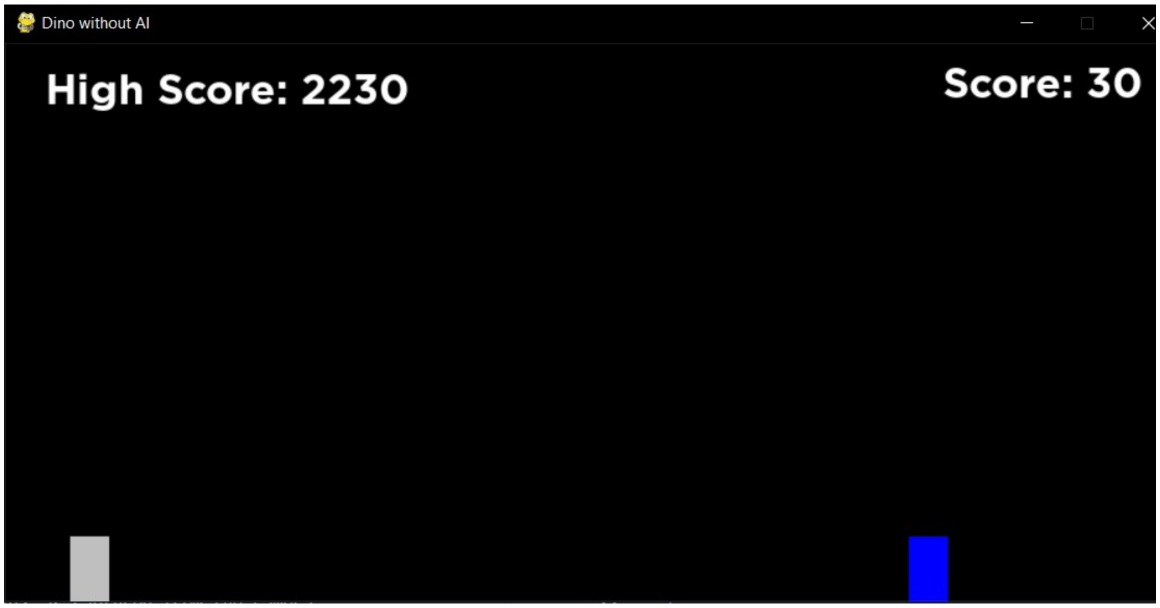
in the game You can jump, dodge obstacles and the character follows your actual body movements as captured from the camera

To do List

- Create The Base Game
    - ☒ Its a game where you have to dodge obstacles
    - ☒ Create the Sprites
    - Create the body models
- Train an AI model to recognise body movement
    - Preferably use Google's teachable machine otherwise we could do tensorflow from scratch (Although that would create a problem with the dataset but whatever)
- Track average body postion relative to previous position to check if it jumped
    - if it did jump then jump the main character
- If possible, add body models to the game along with animations that follow the body movement

Output

Gameplay



Start Screen

Game Over Screen

Main.py Documentation

Imports

```
from pygameInit import *
from myColors import *
from MyFunctions import *
import random, os
from ObjectClass import *
from scene import *
```

- Import necessary modules for the program.

Initialization

```
print("RAAAAAAAAAAAAAH IM STARTING UP RAWr")
```

- Print a startup message.

Custom Exception

```
class myException(Exception):
    def __init__(self, id):
        self.id = id
```

- Define a custom exception class `myException` with an identifier.

Environment Attributes

```
vector = pygame.math.Vector2
width, height = startPygame(hypo = 1000, ratioa = 21, ratiob = 10,
caption = "Dino without AI")
screen = pygame.display.set_mode((width, height))
clock = pygame.time.Clock()
```

- Set up environment attributes such as screen dimensions and clock.

Game Over Function

```
def gameOverFunc(i):
    print("Game Over by Collision of", i.Attributes["id"])
    environmentAttributes["GameOver"] = True
    raise myException("GameOver")
```

- Define a function to handle game over events.

High Score Function

```
def setHighScore():
    with open(highScoreFile, 'w') as file:
        file.write(str(environmentAttributes["highScore"]))
```

- Define a function to set the high score.

Environment Setup

```
font = pygame.font.Font(r"Gotham-Bold.otf", 32)
```

- Set the font for rendering text.

```
environmentAttributes = {...}
```

- Define attributes for the game environment such as gravity, screen dimensions, and score.

Player and Obstacle Attributes

```
playerAttributes = {...}
obstacleAttributes = {...}
```

- Define attributes for the player and obstacles.

Environment Reset

```
def resetEnv():
    ...
```

- Define a function to reset the environment.

High Score File

```
script_dir = os.path.dirname(os.path.abspath(__file__))
highScoreFile = os.path.join(script_dir, 'highScore.txt')
```

- Define the path for the high score file.

```
with open(highScoreFile, 'r') as file:
    environmentAttributes["highScore"] = int(float(file.read()))
```

- Read the high score from the file.

Environment Reset and Initialisation

```
def resetEnv1():
    ...
```

- Define a function to reset the environment and set the high score.

```
resetEnv()
```

- Reset the environment.

Defining Functions

```
def updateEnv():
    ...
```

- Define a function to update the environment.

```
def checkEvent(event):
    ...
```

- Define a function to check events.

```
def centerText(centerTextStr):
    ...
```

- Define a function to center text on the screen.

```
def StartScreen():
    ...
```

- Define a function for the start screen.

```
def GameLoop():
    ...
```

- Define the main game loop function.

```
def GameOver():
    ...
```

- Define a function for the game over screen.

```
def Won():
    ...
```

- Define a function for the win screen.

Scene Management

```
scenes = {...}
```

- Define scenes for managing different parts of the game.

```
currentScene  = "StartScreen"
```

- Set the current scene to the start screen.

```
def changeScene(a):
    ...
```

- Define a function to change scenes.

Main Loop

```
while True:
    ...
```

- **Run the main game loop.**

GameObject.py Documentation

This Part details the the GameObject class ## Imports

```
from MyFunctions import *
import pygame, random, sys
```

- Import necessary modules for the program.

Box Class

```
class Box:
    def __init__(self, Attr, env):
        ...
```

- Define the Box class for game objects.

Attributes:

- `Attributes`: Dictionary containing attributes of the box.
- `environmentAttributes`: Dictionary containing environment attributes.

Methods:

Initialization:

- `__init__(self, Attr, env)`: Initialize the Box object with given attributes and environment.

Main Functions:

- `run(self)`: Run the Box object.
- `update(self)`: Update the position and velocity of the Box object.
- `render(self)`: Render the Box object on the screen.

Collision Handling:

- `checkCollision(self, ObjList)`: Check collision between the Box object and a list of other objects.
- `onCollision(self)`: Define action to be taken upon collision.

Forces and Acceleration:

- `Force(self, F)`: Apply force to the Box object.
- `Force1(self, F, i)`: Apply force in a specific direction to the Box object.

Miscellaneous:

- `debug(self)`: Print debug information about the Box object.

Helper Functions:

- `changeVel(self, i)`: Change the velocity of the Box object.
- `randomScalar(self, i)`: Generate a random scalar value.
- `randomVector(self)`: Generate a random vector.
- `randomise(self, what)`: Randomize specific attributes of the Box object.

Helper Functions

```
vector = pygame.math.Vector2
coll = checkCollisionVector
```

- Define helper functions and variables.