**Documentation for term project**

**Branch: EBS**

**Roll No: CB.EN.P2EBS22005**

**Name: Sravan Kumar N**

**Introduction:**

This script is a basic game written in Python using the Pygame library. The game features a bird that the player controls by moving up and down, obstacles that move towards the bird, collision detection between the bird and obstacles, a scoring system, and a start and game over screen. The game is based on a simple concept of dodging obstacles while collecting points. The game runs in a loop, where the background, bird, and obstacles are continuously redrawn on the screen. The player can control the bird's movement by pressing the spacebar to make the bird move up and releasing it to make the bird fall. The game ends when the bird collides with an obstacle or the user closes the window. The script also keeps track of the player's score and the highest score achieved in previous games, and displays them on the game over screen.

**Modules Used:**

* Random Module
* Pygame Module

**Objective:**

This script is a basic game written in Python using the Pygame library. Pygame is a set of Python modules designed for writing video games. It includes computer graphics and sound libraries that are designed to be used with the Python programming language.

The script starts by importing the necessary modules, initializing Pygame, and setting the display window size.

The script then loads and displays a background image, a bird image, and obstacles on the screen. The bird's position is controlled by user input and the obstacle's position changes automatically. The bird's collision with the obstacle is detected by the collision detection function which checks if the obstacle and bird coordinates overlap.

The script also keeps track of the player's score, and displays the score on the screen.

The script also has a start screen which is displayed when the game first runs and waits for the user to press the space bar to start the game.

The script also has a game over screen which is displayed when the bird collides with an obstacle or the user closes the window. This screen shows the player's score and the highest score achieved in previous games.

The game loop is run while the variable "running" is true, and the script waits for user input to start the game on the start screen and ends the game when there is a collision or when the user exits the window.

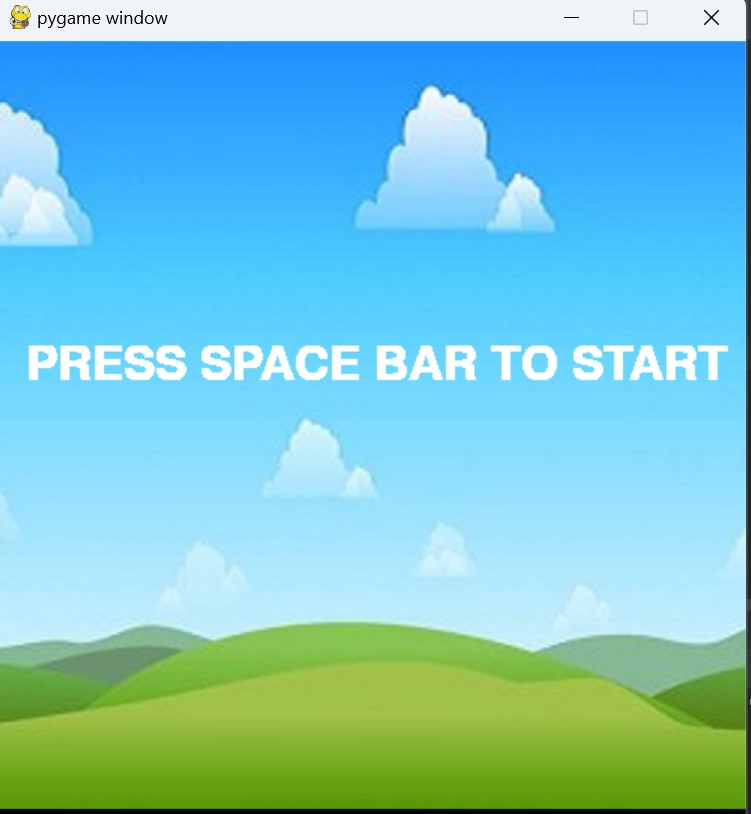
This script is a good starting point for learning game development with Pygame and understanding the basic concepts of game development.

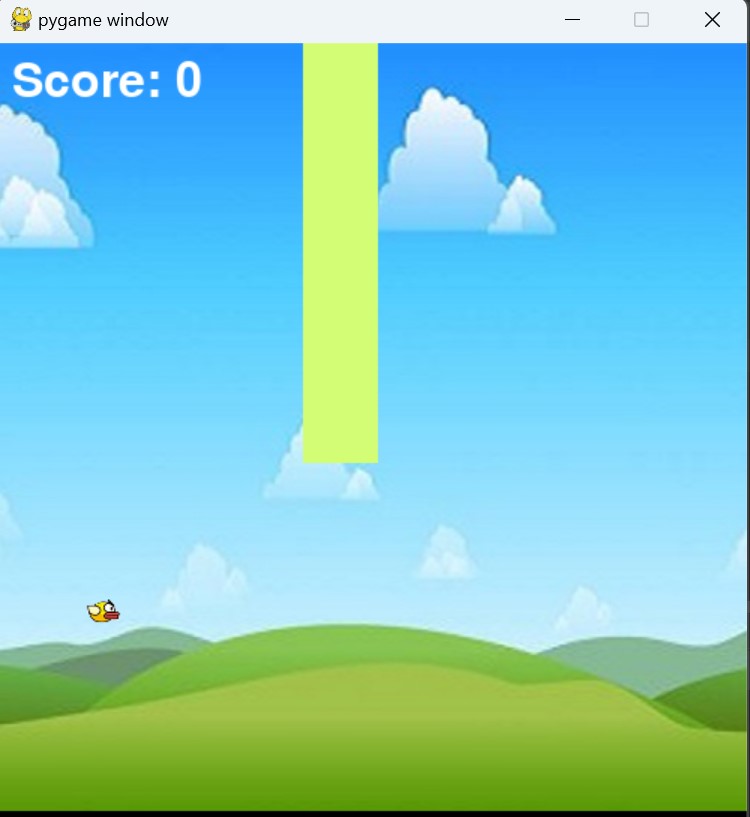
Code:

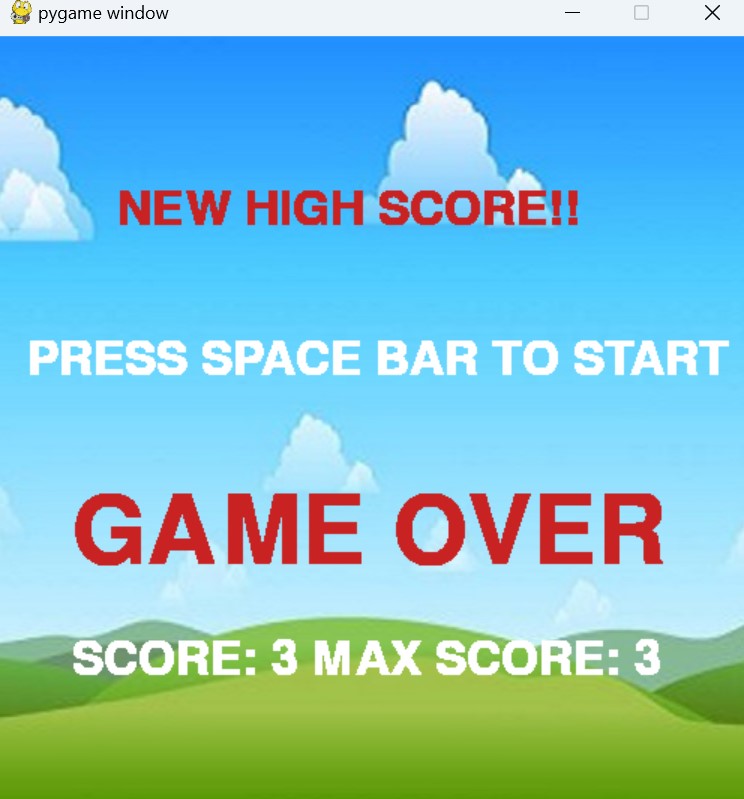
"""  
Author: Sravan Kumar N  
This application is a simple 1-dimension Game which has been created using pygame library  
This file the source code to execute the file and see the output of the game  
"""

import pygame  
import random  
  
# Initialising the modules in pygame  
pygame.init()  
  
SCREEN = pygame.display.set\_mode((500, 550)) # Setting the display  
  
# background  
BACKGROUND\_IMAGE = pygame.image.load('background.jpg')  
  
# BIRD  
BIRD\_IMAGE = pygame.image.load('bird.png')  
bird\_x = 50  
bird\_y = 100  
bird\_y\_change = 0  
  
def display\_bird(x, y):  
 SCREEN.blit(BIRD\_IMAGE, (x, y))  
  
# OBSTACLES  
OBSTACLE\_WIDTH = 50  
OBSTACLE\_HEIGHT = random.randint(150,450)  
OBSTACLE\_COLOR = (211, 253, 117)  
OBSTACE\_X\_CHANGE = -0.1  
obstacle\_x = 500  
  
def display\_obstacle(height):  
 pygame.draw.rect(SCREEN, OBSTACLE\_COLOR, (obstacle\_x, 0, OBSTACLE\_WIDTH, height))  
 bottom\_obstacle\_height = 635 - height - 150  
 pygame.draw.rect(SCREEN, OBSTACLE\_COLOR, (obstacle\_x, 35, OBSTACLE\_WIDTH, -bottom\_obstacle\_height))  
  
# COLLISION DETECTION  
def collision\_detection (obstacle\_x, obstacle\_height, bird\_y, bottom\_obstacle\_height):  
 if obstacle\_x >= 50 and obstacle\_x <= (50 + 64):  
 if bird\_y <= obstacle\_height or bird\_y >= (bottom\_obstacle\_height - 64):  
 return True  
 return False  
  
# SCORE  
score = 0  
SCORE\_FONT = pygame.font.Font('freesansbold.ttf', 32)  
  
def score\_display(score):  
 display = SCORE\_FONT.render(f"Score: {score}", True, (255,255,255))  
 SCREEN.blit(display, (10, 10))  
  
# START SCREEN  
startFont = pygame.font.Font('freesansbold.ttf', 32)  
def start():  
 # displays: "press space bar to start)  
 display = startFont.render(f"PRESS SPACE BAR TO START", True, (255, 255, 255))  
 SCREEN.blit(display, (20, 200))  
 pygame.display.update()  
  
# GAME OVER SCREEN  
# This list will hold all of the scores  
score\_list = [0]  
  
game\_over\_font1 = pygame.font.Font('freesansbold.ttf', 64)  
game\_over\_font2 = pygame.font.Font('freesansbold.ttf', 32)  
  
def game\_over():  
 # check for the maximum score  
 maximum = max(score\_list)  
 # "game over"  
 display1 = game\_over\_font1.render(f"GAME OVER", True, (200,35,35))  
 SCREEN.blit(display1, (50, 300))  
 # shows your current score and your max score  
 display2 = game\_over\_font2.render(f"SCORE: {score} MAX SCORE: {maximum}", True, (255, 255, 255))  
 SCREEN.blit(display2, (50, 400))  
 # If your new score is the same as the maximum then u reached a new high score  
 if score == maximum:  
 display3 = game\_over\_font2.render(f"NEW HIGH SCORE!!", True, (200,35,35))  
 SCREEN.blit(display3, (80, 100))  
  
running = True  
# waiting is going to refer to our end or start screen  
waiting = True  
# set collision to false in the beginning so that we only see the start screen in the beginning  
collision = False  
  
while running:  
  
 SCREEN.fill((0, 0, 0))  
  
 # display the background image  
 SCREEN.blit(BACKGROUND\_IMAGE, (0, 0))  
  
 # we will be sent into this while loop at the beginning and ending of each game  
 while waiting:  
 if collision:  
 # If collision is True (from the second time onwards) we will see both the end screen and the start screen  
 game\_over()  
 start()  
 else:  
 # This refers to the first time the player is starting the game  
 start()  
  
 for event in pygame.event.get():  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_SPACE:  
 # If we press the space bar we will exit out of the waiting while loop and start to play the game  
 # we will also reset some of the variables such as the score and the bird's Y position and the obstacle's starting position  
 score = 0  
 bird\_y = 300  
 obstacle\_x = 500  
 # to exit out of the while loop  
 waiting = False  
  
 if event.type == pygame.QUIT:  
 # in case we exit out make both running and waiting false  
 waiting = False  
 running = False  
  
 for event in pygame.event.get():  
 if event.type == pygame.QUIT:  
 # If you press exit you exit out of the while loop and pygame quits  
 running = False  
  
 if event.type == pygame.KEYDOWN:  
 if event.key == pygame.K\_SPACE:  
 # if you press spacebar you will move up  
 bird\_y\_change = -0.1  
  
 if event.type == pygame.KEYUP:  
 if event.key == pygame.K\_SPACE:  
 # when u release space bar you will move down automatically  
 bird\_y\_change = 0.05  
  
 # moving the bird vertically  
 bird\_y += bird\_y\_change  
 # setting boundaries for the birds movement  
 if bird\_y <= 0:  
 bird\_y = 0  
 if bird\_y >= 571:  
 bird\_y = 571  
  
 # Moving the obstacle  
 obstacle\_x += OBSTACE\_X\_CHANGE  
  
 # COLLISION  
 collision = collision\_detection(obstacle\_x, OBSTACLE\_HEIGHT, bird\_y, OBSTACLE\_HEIGHT + 150)  
  
 if collision:  
 # if a collision does occur we are gonna add that score to our list of scores and make waiting True  
 score\_list.append(score)  
 waiting = True  
  
 # generating new obstacles  
 if obstacle\_x <= -10:  
 obstacle\_x = 500  
 OBSTACLE\_HEIGHT = random.randint(200, 400)  
 score += 1  
 # displaying the obstacle  
 display\_obstacle(OBSTACLE\_HEIGHT)  
  
 # displaying the bird  
 display\_bird(bird\_x, bird\_y)  
  
 # display the score  
 score\_display(score)  
  
 # Update the display after each iteration of the while loop  
 pygame.display.update()  
  
# Quit the program  
pygame.quit()

**Output:**







**Conclusion:**

In conclusion, this script is a basic game written in Python using the Pygame library. It demonstrates the basic concepts of game development such as displaying images, handling user input, and implementing collision detection. The game is simple and easy to understand, making it a good starting point for learning game development with Pygame. The script also includes a scoring system, a start and game over screen, and a high score tracking system. The script can be modified and expanded upon to add more features and improve the overall gameplay experience. Overall, this script serves as a good starting point for anyone looking to learn game development with Pygame and understand the basic concepts of game development.

**Reference:**

* [Flappy Bird Game using PyGame in Python - Javatpoint](https://www.javatpoint.com/flappy-bird-game-using-pygame-in-python)