॥वसुधैव कुटुम्बकम्॥

SYMBIOSIS

INSTITUTE OF TECHNOLOGY

PROGRAMMING AND PROBLEM-SOLVING:

PROJECT-BASED LEARNING

DODGE THE PIPES: MARINE EDITION

A project by:-

Vidhi Binwal (22070122249)

Tanvi Sontakke (22070122234)

Soumili Biswas (22070122216)

Pranavi Singh (22070122211)

INTRODUCTION:

After brainstorming with the group, we conjointly arrived at the conclusion of developing a game which we coined as DODGE THE PIPES – MARINE EDITION, inspired from the highly popular game, FLAPPY BIRD, which is an endless game, coded in Python language that includes a fish being controlled by the user, with the objective of preventing it from colliding with the pipes. Each time the fish passes through the gaps between the pipes, the score increments by one and the game ends when the fish collides with the pipes.
The game is very user-friendly and has an easy-to-understand UI.

The game uses the PyGame library, which mainly comprises of functions and modules needed for game development and can be installed using the PIP installer by typing the following command "pip install pygame" in the command prompt or terminal.
Some of the prerequisites required for this project are given below:

- Python IDLE or Visual Studio Code (IDEs)
- PyGame, Random, and sys modules
- Operating System: Windows 7 and above
- Processor: Intel Core™ i3 or above
- 4GB and above RAM

The code behind our game saw the extensive use of functions, loops and conditions. It is a logical amalgamation of various complex topics. Through this game, we saw the practical application of topics that we had only heard of and studied from a theoretical point of view earlier. The game revolves around a fish that has to avoid all obstacles, in the form of industrial pipes, to keep going and collecting points. Any kind of collision would render the fish dead and the player will have to restart. As simple as it sounds, the coding behind it was definitely not as straight forward. The insertion of sound effects makes the game more realistic and gives us feedback, like the instance of when the fish dies.

The code begins with the import of necessary modules, namely pygame, random and sys. Pygame is an integral part of all game-based programs. It is a cross-platform set of all Python modules designed for writing video games. We have used many functions from this module, for example, - colliderect(), .get_rect(), .blit() etc.

The random module was used to generate pipes randomly. The positions of the pipes keep changing randomly; this is the work of the random.choice() function. These are pseudo-random numbers, implying that these are not truly random.

The sys module in Python provides various functions and variables that are used to manipulate different parts of the Python runtime environment. It allows operating on the interpreter as it provides access to the variables and functions that interact strongly with the interpreter. The examples of its use in our program specifically include - sysfont(), sys.exit().

Functions are probably the most important parts of our code. We had to create many different functions, catering to varied requirements, while using some built-in ones too. They are as concise as possible, and our code would be insignificant without these utilitarian functions.

<u>ALGORITHM:</u>

Step 1: Start

Step 2: Display the "Get Ready" and "Press Space" messages.

Step 3: Check whether GameActive condition is True or not.

Step 4: If yes, check if the condition is Quit or KeyDown.
In case of Quit, break the loop and exit it.
In case of KeyDown, initiate the game and the concomitant loops required to run it.

Step 5: Collect points and keep the fish moving and poles moving as long as isalive and isstarted are True. Also, check for collisions.

Step 6: Play bubble_sound while the fish is moving and die_sound on collision with any pipe.

Step 7: On collision, reset the position of the fish, update the high score and reset the initial score to 0. Also, display the “Get Ready” and “Press Space” messages again.

Step 8: If user presses space, restart the game. If user quits, exit the game window.

Step 9: Stop

CODE:

```
#IMPORTING REQUIRED MODULES
import pygame
import random
import sys


# creating the game floor (seabed)
def game_floor():
    screen.blit(floor, (floorX, 590))
    screen.blit(floor, (floorX+565, 590)) # keeps adding a new game floor
after each new x coordinate



# check collision with pipes
def check_collision(pipes):
    # collision with pipes
    for pipe in pipes:
        if fish_rect.colliderect(pipe):
            die_sound.play()
            return False


    # check floor and screen top is not hit
    if (fish_rect.top <= 0 or fish_rect.bottom >= 600):
        die_sound.play()
        return False
    return True

#updates the score counter with the new value and stores it
def update_score(score, high_score):
    if score > high_score:
        high_score = score
    return high_score

#randomly generates pipes
def create_pipe():
    random_pipe_pos = random.choice(pipe_height)#choose any of the
heights provided
    top_pipe = pipe_surface.get_rect(midbottom=(1000,
random_pipe_pos-300))#1000 gives the illusion of FPS
    bottom_pipe = pipe_surface.get_rect(midtop=(1000,
random_pipe_pos))
    return bottom_pipe, top_pipe

#the pipe is being generated at each pixel, which gives an illusion of it
being moving
```

```
def move_pipes(pipes):
    for pipe in pipes:
        pipe.centerx -= 5
    return pipes

#creating pipes
def draw_pipes(pipes):
    for pipe in pipes:
        if pipe.bottom >= 690:
            screen.blit(pipe_surface, pipe)
        else:
            flip_pipe = pygame.transform.flip(pipe_surface, False, True)
            screen.blit(flip_pipe, pipe)



#initialize pygame
pygame.init()
clock = pygame.time.Clock()
fish_move = 0
GAME_FONT = pygame.font.SysFont('Comic Sans MS', 28)#selects the
font for the game
gravity = 0.25
```

```
color = (255, 000, 000)#sets the text color

# Create pygame window
screen = pygame.display.set_mode((565, 690))

# to add an icon and title to the window
pygame.display.set_caption('DODGE THE PIPES - MARINE EDITION')
icon = pygame.image.load('icon.png')
pygame.display.set_icon(icon)

# load background
background1 = pygame.image.load('background1.png').convert()
background1 = pygame.transform.scale2x(background1)

# to load the floor
floor = pygame.image.load('gamefloor2.png').convert()
floor = pygame.transform.scale2x(floor)
floorX = 0  # current position of game floor, which will be recreated with
every while loop

# to load player (fish)
# we use convert_alpha for transparent background images
```

```
fish = pygame.image.load('fish.png').convert_alpha()
fish_rect = fish.get_rect(center=(100, 345))

# to load get ready message
message1 = pygame.image.load('message1.png').convert_alpha()
start_game_rect = message1.get_rect(center=(282, 150))

# to load replay message
replay = pygame.image.load('replay.png').convert_alpha()
replay_rect = replay.get_rect(center=(282, 450))

# to load score image
score_img = pygame.image.load('score_img.png').convert_alpha()
score_img_rect = score_img.get_rect(center=(50, 30))

# to load high score img:
hscore = pygame.image.load('hscore.png').convert_alpha()
hscore_rect = hscore.get_rect(center=(50, 80))

# building pipes
pipe_surface = pygame.image.load('tube2.png')
pipe_list = []
```

```
pipe_height = [400, 500, 600]#a list of pipe heights, which the program
automatically selects

#SPAWNPIPE is a variable, basically creates a new pipe
SPAWNPIPE = pygame.USEREVENT
pygame.time.set_timer(SPAWNPIPE, 1200)#1200=FPS, .set_timer() asks
the program to create a new pipe every 1.2 sec

# background sounds
die_sound = pygame.mixer.Sound('sfx_die.mp3')
bubble_sound=pygame.mixer.Sound('sfx_bubble.mp3')

# create infinite while loop (main game)
game_active = True
isstarted = False
isalive = False
scorectr = 0
flag = 0
high_score = 0

while True:
    for event in pygame.event.get():#for every event in the pygame
window
```

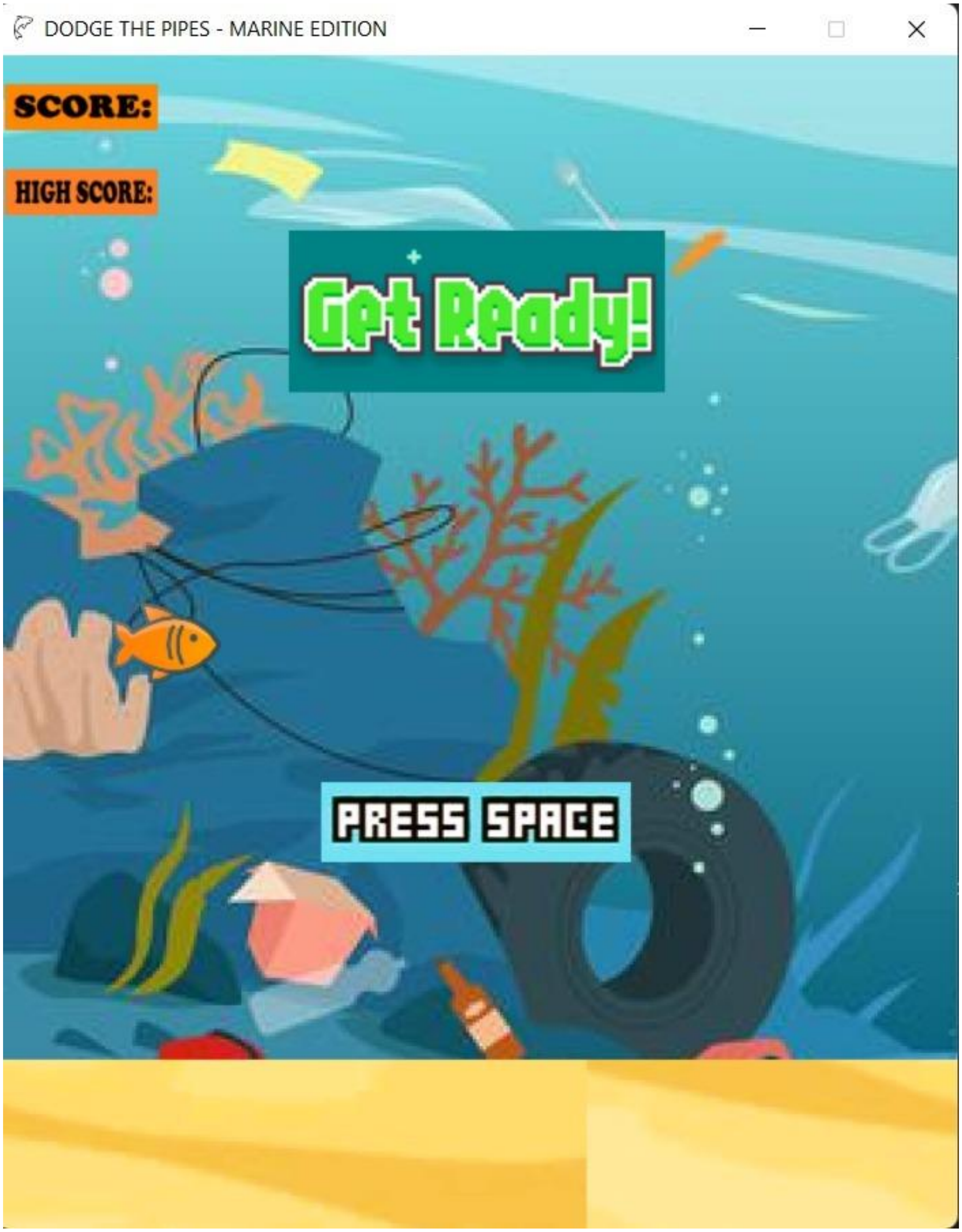
```
    if event.type == pygame.QUIT or (event.type == pygame.KEYDOWN
and event.key == pygame.K_ESCAPE): #if the type of event is the process
of Quitting
        pygame.quit()
        sys.exit()


    if event.type == SPAWNPIPE and game_active: #the type of event is
a pipe being created, extend the list of pipes
        pipe_list.extend(create_pipe())


    if event.type == pygame.KEYDOWN: #if the type of event is the
process of pressing down a key
        if event.key == pygame.K_SPACE and not isstarted: #if the key
pressed is the spacebar, let isalive and isstarted be true (indicates that
the fish is alive and stationary)
            isstarted = True
            isalive = True
        if event.key == pygame.K_SPACE and isstarted and isalive: #if the
spacebar is pressed, move the fish up 9 pixels
            fish_move = 0
            fish_move -= 9 #gives the illusion of the fish jumping up 9
pixels, when its being recreated at each pixel very fast
            bubble_sound.play()
        elif event.key == pygame.K_SPACE and not isalive: #if the fish hits
the pipe (isalive becomes false), let the score=0
```

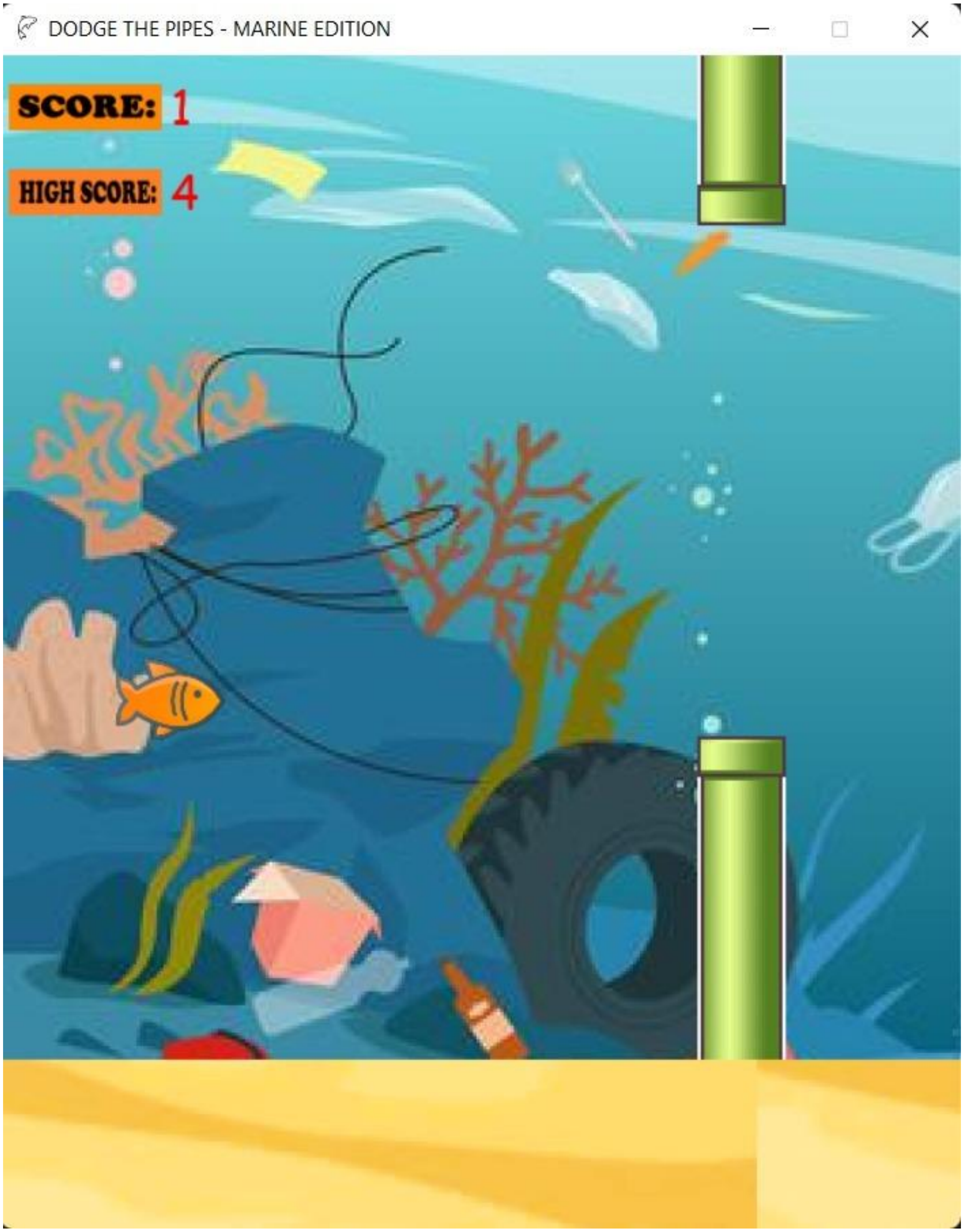
```
            # print("restar")

            isalive = True

            game_active = True

            scorectr = 0


   screen.blit(background1, (0, -10)) #creating the background

   if pipe_list != [] and isstarted and isalive: #if the pipe list is not empty,
the fish is stationary, and alive

       for i in pipe_list: #for each element in the pipe list

           flag = 1 #let this parameter be 1 (true)

           if i[0] < 0: #if the index of the element of the list is < 0

               pipe_list.pop(pipe_list.index(i)) # pop that element from the
list, and clear the list every time its updated

               scorectr = scorectr+1 # increase the score by 1

       else:

           if flag:

               flag = 0


   if game_active and isstarted == True:

       fish_move += gravity #let the fish drop by 0.25 pixels

       fish_rect.centery += fish_move  # to detect any collision around the
fish
```

```
        text = GAME_FONT.render(str(scorectr//2), True, color) # score text
variable (divide by 2)
        text1 = GAME_FONT.render(str(high_score//2), True, color) #
highscore text variable
        screen.blit(text, (100, 10)) # print the score
        screen.blit(text1, (100, 60)) # print the highscore
        # draw pipes
        pipe_list = move_pipes(pipe_list)
        draw_pipes(pipe_list)

        # check for collision
        game_active = check_collision(pipe_list)
        isalive = game_active

    else:
        # the game has ended, after collision
        # to detect any collision around the fish
        fish_rect.center = (100, 345)
        fish_move = 0
        pipe_list.clear()
        game_activate = True
        high_score = update_score(scorectr, high_score) # update the high
score with the current score
```

```
        screen.blit(message1, start_game_rect) # display "GET READY" msg
        screen.blit(replay, replay_rect) # display "PRESS SPACE" msg
    screen.blit(fish, fish_rect) # draw the fish sprite
    screen.blit(score_img, score_img_rect) # draw the "SCORE" img
    screen.blit(hscore, hscore_rect) # draw the "HIGH SCORE" img


    # create floor
    floorX -= 1
    game_floor()
    # if the value of the x coordinate of the floor becomes less than 565
(which is the width of the game window) x coordinate becomes 0
    if floorX <= -565:
        # let it become 0 (stop the loop) if the x coordinate is becomes less
than the width of the window (which is not possible), thus, creates an
infinite loop
        floorX = 0


    pygame.display.update() # keeps updating the pygame screen


    clock.tick(100)
```

OUTPUT:


DODGE THE PIPES - MARINE EDITION
SCORE:
HIGH SCORE:
Get Ready!
PRESS SPACE


DODGE THE PIPES - MARINE EDITION
SCORE: 1
HIGH SCORE: 4

CONCLUSION / WHAT WE UNDERSTOOD:

In conclusion, we learnt a lot while making the game. To start with, we were introduced to the world of game development and acquired knowledge about various libraries and modules along with their implementation and installation in Python, pertaining specifically to game development.
The code behind our game saw the extensive use of functions, loops, and conditions. It is a logical amalgamation of various complex topics. Through this game, we saw the practical application of the topics we had only heard of and studied from a theoretical point of view.
We thank our PPS faculty for giving us the opportunity to work in a team, learn and explore the realm of game development by using optimized Python codes, making flowcharts, algorithms, and using new libraries and modules which inculcated an interest in us to develop games in the future.

We would like to end this project by stating that:

Change in something significant needs an innovative approach which entertains as well as enlightens the mind otherwise oblivious to the concerning issues.

REFERENCES:

https://copyassignment.com/flappy-bird-in-python-pygame-with-source-code/

https://www.youtube.com/watch?v=FfWpgLFMI7w

https://www.youtube.com/watch?v=_7er9kqWpG4

https://www.javatpoint.com/flappy-bird-game-using-pygame-in-python