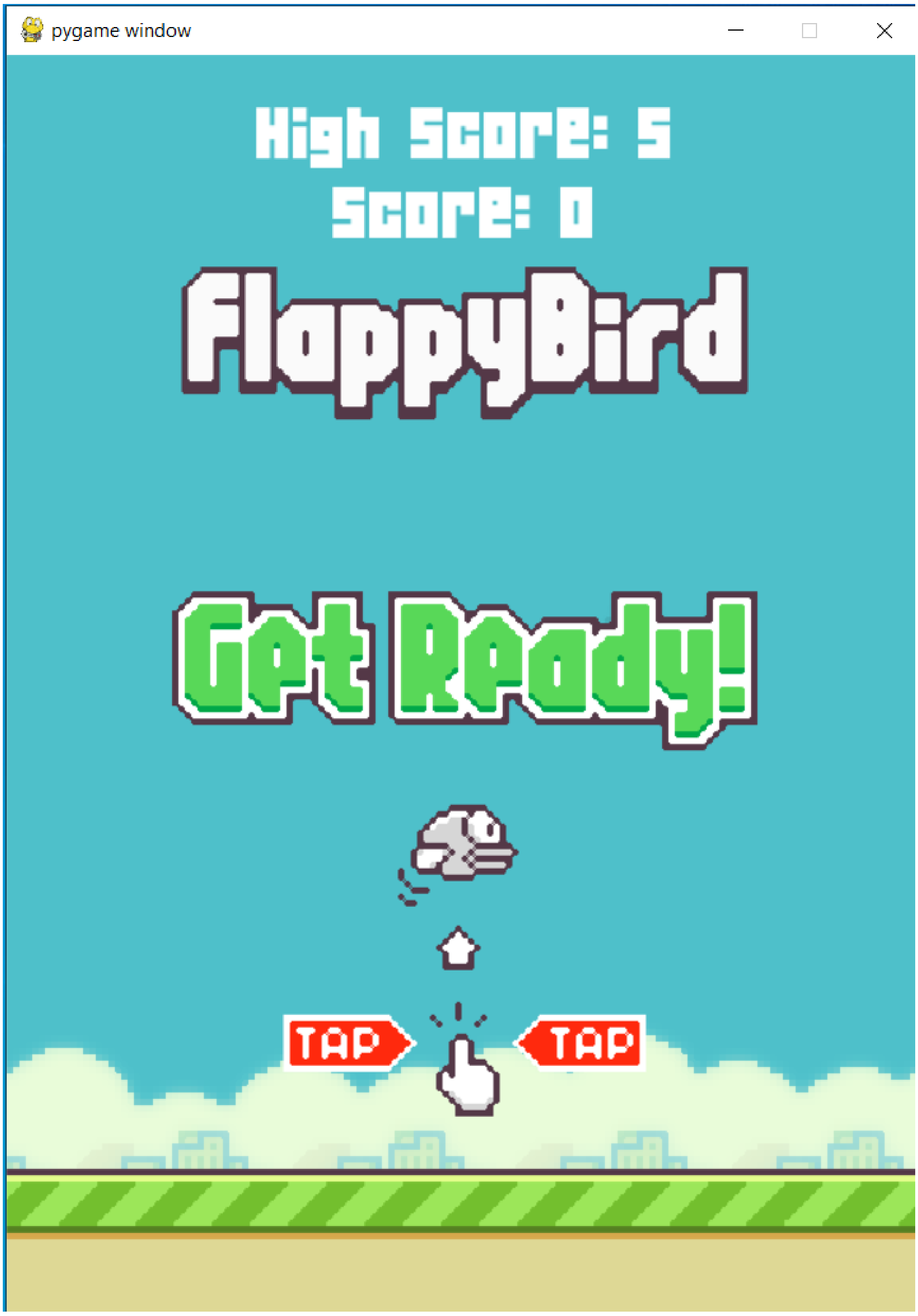
FLAPPY BIRD


pygame window
High Score: 5
Score: 0
FlappyBird
Get Ready!
TAP
TAP

<u>INTRODUCTION:</u>

The “Flappy Bird” is an arcade-style game in which the player controls the blue bird, which moves persistently to the right. The game starts once we press the space button. The players are tasked with navigating the bird through the pipes that have equally sized gaps placed at random heights. The blue bird automatically descends and only ascends when the player presses space bar. Each successful pass through a pair of pipes awards the player with one point. The display also shows the Score and the High scores after each attempt.

<u>MAIN FEATURE:</u>

<u>1.LIBRARIES USED:</u>

So, the game needs to have three basic libraries so as to play with the same number of graphics we used to play in mobile. The libraries are pygame, random and sys. So pygame would help us to set the display, add backgrounds and sound effects. Whereas the random library is used to set different heights of pipes during the game.

<u>2.KEY CONTROLS:</u>

The ascending of the blue bird is done by pressing the space bar. This is done by pygame.KEYDOWN and in specific pygame.K_SPACE.

<u>3.DISPLAY OF BACKGROUND SCREEN:</u>

To set the background screen, I used pygame.display.set_mode((576,800)). The tuple value signifies the dimensions of the screen.

4.BACKGROUND IMAGES AND SOUNDS:

So, there are several images loaded on to this program to execute this game like the bird, day background, message like Score, High score and then the pipes and the base. This is done by pygame.image.load. And further, to add to the background display, I used screen.blit(). Other than this, there are several sound effects added when the bird ascends, collides and points. This is done by pygame.mixer.

5. MOVEMENT OF THE BIRD:

So, there are three types of movements for the bird I have loaded in this program- up-flap, mid-flap, down-flap. This is done by clock.tick().

6.ANIMATION OF THE FLOOR AND SPAWNING OF PIPES:

So, for this, I have created a function in which the x-axis i.e., is the length of the display keeps changing while the y-axis remains the same. Now to create an animation of the floor, I ran a condition in which after the floor goes till the end of the length in -x-axis, it will reset to 0 and the function gets executed again and again till we lose. Same is the case for pipes, but it should look like that it is coming from the right. So, I increased the coordinates of x-axis, so that it will look like it is coming from the right.

THINGS THAT I LEARNT:

1. I learnt how to use pygame for getting images and sound effects into our program.
2. I learnt how to set a background display screen.
3. I learnt basic animation through the animation of floor and respawning of the pipes.

BASECODE AND REFERENCES:

1. https://www.youtube.com/watch?v=UZg49z76cLw&t=908s
2. https://www.youtube.com/watch?v=i6xMBig-pP4

CODE:

```
import pygame, sys, random


def animation_floor():
    screen.blit(floor_surface, (floor_x_pos, 700))
    screen.blit(floor_surface, (floor_x_pos + 576, 700))


def create_pipe():
    random_pipe_pos = random.choice(pipe_height)
    bottom_pipe = pipe_surface.get_rect(midtop=(700, random_pipe_pos))
    top_pipe = pipe_surface.get_rect(midbottom=(700, random_pipe_pos -
300))
    return bottom_pipe, top_pipe


def move_pipes(pipes):
    for pipe in pipes:
        pipe.centerx -= 5
    visible_pipes = [pipe for pipe in pipes if pipe.right > -50]
    return visible_pipes


def draw_pipes(pipes):
    for pipe in pipes:
        if pipe.bottom >= 800:
            screen.blit(pipe_surface, pipe)
        else:
            flip_pipe = pygame.transform.flip(pipe_surface, False, True)
            screen.blit(flip_pipe, pipe)


def check_collision(pipes):
    global can_score
    for pipe in pipes:
        if bird_rect.colliderect(pipe):
            death_sound.play()
            can_score = True
            return False

    if bird_rect.top <= -100 or bird_rect.bottom >= 700:
        can_score = True
        return False

    return True
```

```python
def rotate_bird(bird):
    new_bird = pygame.transform.rotozoom(bird, -bird_movement * 3, 1)
    return new_bird


def bird_animation():
    new_bird = bird_frames[bird_index]
    new_bird_rect = new_bird.get_rect(center=(100, bird_rect.centery))
    return new_bird, new_bird_rect


def score_display(game_state):
    if game_state == 'main_game':
        score_surface = game_font.render(str(int(score)), True, (255, 255,
255))
        score_rect = score_surface.get_rect(center=(288, 100))
        screen.blit(score_surface, score_rect)
    if game_state == 'game_over':
        score_surface = game_font.render(f'Score: {int(score)}', True,
(255, 255, 255))
        score_rect = score_surface.get_rect(center=(288, 100))
        screen.blit(score_surface, score_rect)

        high_score_surface = game_font.render(f'High score:
{int(high_score)}', True, (255, 255, 255))
        high_score_rect = high_score_surface.get_rect(center=(288, 50))
        screen.blit(high_score_surface, high_score_rect)


def update_score(score, high_score):
    if score > high_score:
        high_score = score
    return high_score


def pipe_score_check():
    global score, can_score

    if pipe_list:
        for pipe in pipe_list:
            if 95 < pipe.centerx < 105 and can_score:
                score += 1
                score_sound.play()
                can_score = False
            if pipe.centerx < 0:
                can_score = True


# pygame.mixer.pre_init(frequency = 44100, size = 16, channels = 2, buffer
= 1024)
pygame.init()
screen = pygame.display.set_mode((576, 800))
clock = pygame.time.Clock()
game_font = pygame.font.Font('04B_19.ttf', 40)

# Game Variables
gravity = 0.3
bird_movement = 0
game_active = True
score = 0
```

```
high_score = 0
can_score = True
bg_surface = pygame.image.load('background-day.png').convert()
bg_surface = pygame.transform.scale2x(bg_surface)

floor_surface = pygame.image.load('base.png').convert()
floor_surface = pygame.transform.scale2x(floor_surface)
floor_x_pos = 0

bird_downflap = pygame.transform.scale2x(pygame.image.load('bluebird-
downflap.png').convert_alpha())
bird_midflap = pygame.transform.scale2x(pygame.image.load('bluebird-
midflap.png').convert_alpha())
bird_upflap = pygame.transform.scale2x(pygame.image.load('bluebird-
upflap.png').convert_alpha())
bird_frames = [bird_downflap, bird_midflap, bird_upflap]
bird_index = 0
bird_surface = bird_frames[bird_index]
bird_rect = bird_surface.get_rect(center=(100, 400))

BIRDFLAP = pygame.USEREVENT + 1
pygame.time.set_timer(BIRDFLAP, 200)


pipe_surface = pygame.image.load('pipe-green.png')
pipe_surface = pygame.transform.scale2x(pipe_surface)
pipe_list = []
SPAWNPIPE = pygame.USEREVENT
pygame.time.set_timer(SPAWNPIPE, 1500)
pipe_height = [400, 500, 600]

game_over_surface =
pygame.transform.scale2x(pygame.image.load('message.png').convert_alpha())
game_over_rect = game_over_surface.get_rect(center=(288, 400))

flap_sound = pygame.mixer.Sound('sfx_wing.wav')
death_sound = pygame.mixer.Sound('sfx_hit.wav')
score_sound = pygame.mixer.Sound('sfx_point.wav')
score_sound_countdown = 100
SCOREEVENT = pygame.USEREVENT + 2
pygame.time.set_timer(SCOREEVENT, 100)

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_SPACE and game_active:
                bird_movement = 0
                bird_movement -= 12
                flap_sound.play()
            if event.key == pygame.K_SPACE and game_active == False:
                game_active = True
                pipe_list.clear()
                bird_rect.center = (100, 400)
                bird_movement = 0
                score = 0

        if event.type == SPAWNPIPE:
            pipe_list.extend(create_pipe())
        if event.type == BIRDFLAP:
            if bird_index < 2:
                bird_index += 1
            else:
                bird_index = 0

            bird_surface, bird_rect = bird_animation()

    screen.blit(bg_surface, (0, 0))

    if game_active:
        # Bird
        bird_movement += gravity
        rotated_bird = rotate_bird(bird_surface)
        bird_rect.centery += bird_movement
        screen.blit(rotated_bird, bird_rect)
        game_active = check_collision(pipe_list)

        # Pipes
        pipe_list = move_pipes(pipe_list)
        draw_pipes(pipe_list)

        # Score
        pipe_score_check()
        score_display('main_game')
    else:
        screen.blit(game_over_surface, game_over_rect)
        high_score = update_score(score, high_score)
        score_display('game_over')

    # Floor
    floor_x_pos -= 1
    animation_floor()
    if floor_x_pos <= -576:
        floor_x_pos = 0

    pygame.display.update()
    clock.tick(60)
```