BACSE101 Problem Solving using Python

PROJECT REPORT

on

Flappy bird

Prepared by

Pranjal Banchhor- 25BCE2024

Under the supervision of

Thirumoorthy Krishnan

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering
Vellore Institute of Technology, Vellore.

November 08, 2025

Table of Contents

Abstract

1. Introduction

2. Problem Statement and Objectives

3. Implementation Code

4. Demo Screenshots

5. Conclusion

Abstract

*This project is a recreation of the classic Flappy Bird game developed using **Python** and the **Pygame** library. The main objective of the game is to control a bird that continuously moves forward and must avoid colliding with oncoming pipes by flapping to stay in the air.*

*The project demonstrates the fundamental concepts of **game development**, including animation, event handling, collision detection, and score tracking. The bird's movement is governed by basic physics — gravity and upward thrust when the player presses the spacebar. Randomly generated pipe obstacles create dynamic gameplay, while a continuously updated score encourages competition.*

The game window displays the current score and provides a restart option upon collision, allowing smooth replay ability. This project highlights efficient use of Pygame for rendering graphics, managing frame rates, and handling user input to create an engaging, interactive gaming experience.

1. Introduction

1.1 This project is a Python-based recreation of the popular *Flappy Bird* game using the Pygame library. The objective of the game is to control a bird that must navigate through an endless series of pipes without colliding with them. The game demonstrates key principles of 2D game design, such as real-time rendering, collision detection, event handling, and simple physics-based motion. It provides an engaging way to understand how games manage player interaction, object movement, and scoring mechanisms in a dynamic environment.

1.2 Domain Information

This project falls under the domain of **Game Development** and **Interactive Multimedia Applications**. Game development involves creating visually interactive applications that respond to user inputs and simulate realistic or imaginative environments. The project focuses on 2D physics, object animation, and procedural content generation, all of which are core aspects of casual mobile and desktop gaming.

1.3 Software Libraries Used

- **Python:** The primary programming language used for logic implementation.

- **Pygame:** A Python library designed for creating video games. It handles graphics rendering, sound, events, and input control efficiently.

- **Random (Python Standard Library):** Used for generating random pipe positions to make the gameplay dynamic and unpredictable.

- **Sys (Python Standard Library):** Used for handling program exit and system-related functions.

1.4 Contributions by Team Members

This project was entirely developed by a single individual, who handled all aspects of the design and implementation. The contributions include coding the game logic, implementing bird physics and gravity, designing the graphical interface, managing pipe generation and collisions, and testing the final gameplay.

1.5 Challenges Faced

The main challenge faced while making this project was to perfectly implement the random pipe generation and its looping. Followed by some other problems such as drawing the window which included colouring the pipes, making the bird(circle), etc.

2. Problem Statement and Objectives

2.1 Problem Statement

The aim of this project is to design and develop a simple yet engaging 2D game that replicates the core mechanics of the popular *Flappy Bird* game using **Python** and **Pygame**. The problem lies in creating a smooth, interactive gameplay experience where the player controls a bird that must continuously navigate through randomly generated obstacles (pipes) without colliding with them or the screen boundaries.

The challenge is to simulate realistic motion physics, such as gravity and upward flapping, while maintaining consistent frame rates, collision accuracy, and visually appealing gameplay. Additionally, the game should include score tracking, restarting capability, and responsive controls to ensure user satisfaction and replay ability.

2.2 Objectives

The main objectives of this project are:

1. **To design and implement** a 2D interactive game using the Pygame library.
2. **To simulate** real-world physics effects such as gravity and jump mechanics for smooth bird movement.
3. **To generate** dynamic obstacles (pipes) at random intervals and positions to increase gameplay difficulty.
4. **To detect** collisions accurately between the bird and obstacles or screen boundaries.
5. **To maintain** a scoring system that updates in real-time as the player passes obstacles.
6. **To implement** a restart feature allowing the player to replay the game after losing.
7. **To ensure** efficient frame rendering and responsive controls for a seamless user experience.

3. Implementation

3.1 Feature

The main features implemented in the project are as follows:

1. Bird movement and gravity simulation
2. Dynamic pipe generation and movement
3. Collision detection system
4. Scoring mechanism
5. Game over and restart functionality

Each feature is described and coded below as a separate module-like section.

3.2 Bird Movement and Gravity Simulation

Description:
The bird's movement is controlled by gravity and flap input. Gravity continuously pulls the bird downward, while pressing the spacebar provides upward thrust. The bird's position is updated frame-by-frame for smooth animation

```
if not game_over:
    bird_velocity += gravity
    bird_y += bird_velocity
```

3.3 Dynamic Pipe Generation and Movement

Description:
Pipes are generated at random heights and move continuously from right to left across the screen. When a pipe moves off-screen, it is removed to maintain performance.

```
def create_pipe():
    y_top = random.randint(100, HEIGHT - 200)
    top_rect = pygame.Rect(WIDTH, 0, pipe_width, y_top)
    bottom_rect = pygame.Rect(WIDTH, y_top + pipe_gap, pipe_width, HEIGHT)
    pipes.append((top_rect, bottom_rect))
```

3.4 Collision Detection System

Description:
The game checks if the bird collides with any pipe or touches the top/bottom of the window. If a collision occurs, the game enters the “Game Over” state.

```
# Collision detection

for top, bottom in pipes:

    if top.collidepoint(bird_x, bird_y) or bottom.collidepoint(bird_x, bird_y):

        game_over = True



# Check floor and ceiling

if bird_y - bird_radius <= 0 or bird_y + bird_radius >= HEIGHT:

    game_over = True
```

3.5 Scoring Mechanism

Description:
The player’s score increases by one point each time the bird successfully passes a set of pipes. The score is displayed in real-time on the game window.

```
# Score update

if top.x + pipe_width == bird_x:

    score += 1



# Display score

text = font.render(f'Score: {score}", True, WHITE)

win.blit(text, (10, 10))
```

3.6 Game Over and Restart Functionality

Description:
When the bird collides or falls, a “Game Over” message appears. The player can restart the game by pressing the spacebar, which resets all variables.

```
def reset_game():

    global bird_y, bird_velocity, pipes, score, game_over

    bird_y = HEIGHT // 2

    bird_velocity = 0
```

```
    pipes = []
    score = 0
    game_over = False


# Restart on spacebar
if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_SPACE:
        if game_over:
            reset_game()
        else:
            bird_velocity = flap_strength
```

3.7 Menu Option (Feature Navigation)

Description:
the game structure allows you to leave and enter at will with vibrant, discrete and engaging colouring on its menu.

1. Start Game
2. View Instructions
3. Exit

```
def main_menu():
    menu_font = pygame.font.SysFont("comicsans", 50)
    win.fill(BLUE)
    title = menu_font.render("Flappy Bird", True, WHITE)
    start_text = font.render("Press SPACE to Start", True, WHITE)
    win.blit(title, (WIDTH//2 - title.get_width()//2, HEIGHT//2 - 60))
    win.blit(start_text, (WIDTH//2 - start_text.get_width()//2, HEIGHT//2))
    pygame.display.update()
    main_menu()
```

4. Demo Screenshots

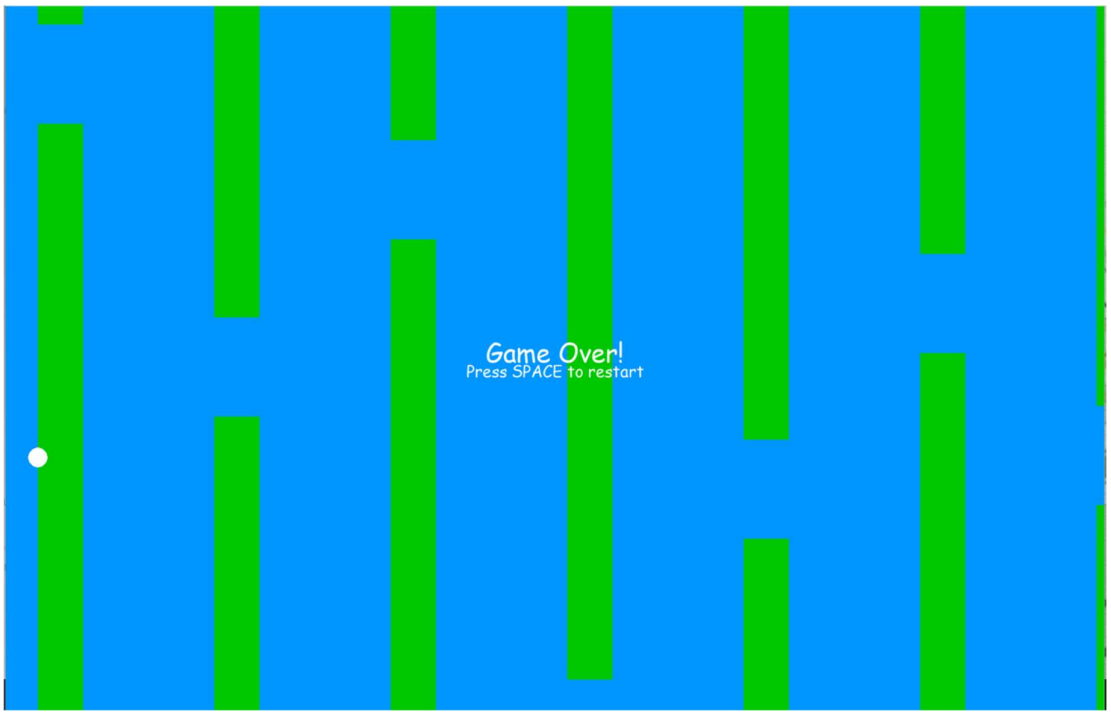
code:

```
import pygame
import random
import sys

# Initialize pygame
pygame.init()

# Game window
WIDTH, HEIGHT = 400, 600
win = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Flappy Bird")

# Colors
WHITE = (255, 255, 255)
GREEN = (0, 200, 0)
BLUE = (0, 150, 255)

# Fonts
font = pygame.font.SysFont("comicsans", 40)

# Game clock
clock = pygame.time.Clock()
FPS = 60

# Load bird image
bird_img = pygame.image.load("bird.png").convert_alpha()
bird_img = pygame.transform.scale(bird_img, (40, 30))  # resize to fit

# Bird
bird_x = 50
bird_y = HEIGHT // 2
bird_velocity = 0
gravity = 0.5
flap_strength = -8

# Pipes
pipe_width = 70
pipe_gap = 150
pipe_speed = 3
pipes = []

# Score
score = 0

# Game states
game_over = False

def reset_game():
    global bird_y, bird_velocity, pipes, score, game_over
    bird_y = HEIGHT // 2
```

```
    bird_velocity = 0
    pipes = []
    score = 0
    game_over = False

def create_pipe():
    """Create a new pipe pair with a random gap position."""
    y_top = random.randint(100, HEIGHT - 200)
    top_rect = pygame.Rect(WIDTH, 0, pipe_width, y_top)
    bottom_rect = pygame.Rect(WIDTH, y_top + pipe_gap, pipe_width, HEIGHT)
    pipes.append((top_rect, bottom_rect))

def draw_window():
    win.fill(BLUE)

    # Draw pipes
    for top, bottom in pipes:
        pygame.draw.rect(win, GREEN, top)
        pygame.draw.rect(win, GREEN, bottom)

    # Draw bird with rotation
    angle = -bird_velocity * 2  # tilt upward when flapping
    rotated_bird = pygame.transform.rotate(bird_img, angle)
    win.blit(rotated_bird, (bird_x - rotated_bird.get_width() // 2, bird_y -
rotated_bird.get_height() // 2))

    # Draw score
    text = font.render(f"Score: {score}", True, WHITE)
    win.blit(text, (10, 10))

    # Draw game over message
    if game_over:
        over_text = font.render("Game Over!", True, WHITE)
        restart_text = pygame.font.SysFont("comicsans", 25).render("Press SPACE to
restart", True, WHITE)
        win.blit(over_text, (WIDTH//2 - over_text.get_width()//2, HEIGHT//2 - 40))
        win.blit(restart_text, (WIDTH//2 - restart_text.get_width()//2, HEIGHT//2))

    pygame.display.update()

# Main loop
pipe_timer = 0
running = True
while running:
    clock.tick(FPS)

    # Input
    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:
                if game_over:
                    reset_game()
                else:
```

```
                bird_velocity = flap_strength

    if not game_over:
        # Bird physics
        bird_velocity += gravity
        bird_y += bird_velocity

        # Pipe generation
        pipe_timer += 1
        if pipe_timer > 90:
            create_pipe()
            pipe_timer = 0

        # Move pipes and check collisions
        new_pipes = []
        for top, bottom in pipes:
            top.x -= pipe_speed
            bottom.x -= pipe_speed

            # Check collision (approximate bounding box)
            bird_rect = pygame.Rect(bird_x - 20, bird_y - 15, 40, 30)
            if top.colliderect(bird_rect) or bottom.colliderect(bird_rect):
                game_over = True

            # Update score
            if top.x + pipe_width == bird_x:
                score += 1

            if top.x + pipe_width > 0:
                new_pipes.append((top, bottom))
        pipes = new_pipes

        # Check floor/ceiling
        if bird_y <= 0 or bird_y >= HEIGHT - 10:
            game_over = True

    # Draw everything
    draw_window()
```

Gameplay picture



5. Conclusion

The *Flappy Bird* project successfully demonstrates the fundamentals of **2D game development** using **Python** and the **Pygame** library. Through this project, important game programming concepts such as event handling, collision detection, frame updates, and physics simulation were effectively implemented.

The game provides an engaging and interactive experience, showcasing how simple logic and design can result in enjoyable gameplay. By integrating random obstacle generation and responsive controls, the project achieves dynamic behaviour and replay ability.

This project not only enhanced understanding of Python's graphical capabilities but also strengthened problem-solving skills, logical thinking, and modular programming practices. The completed game can serve as a foundation for future enhancements, such as adding sound effects, animated sprites, difficulty levels, and main menu options.

In conclusion, the development of *Flappy Bird* using Pygame successfully fulfills its objectives by combining creativity, programming logic, and interactivity into a cohesive and functional game application.