Project Report: Flappy Bird Game

1. Introduction

This project is based on the development of a simplified version of the popular mobile game "Flappy Bird." The objective of the game is to control a bird and navigate it through a series of pipes without colliding with them. The game is developed using the Python programming language and utilizes the **Pygame** library for graphical rendering and handling game events. This report details the design, development process, and implementation of the game, along with insights into how Python and Pygame were utilized to bring the game to life.

2. Objective

The primary objective of this project is to create a functional, engaging, and interactive Flappy Bird game. Players control the bird by clicking the mouse or pressing a key to make it "flap" and avoid obstacles (pipes) that appear on the screen. The aim is to keep the bird in the air for as long as possible, avoiding collisions with the pipes or the ground. The game ends when the bird hits an obstacle, and the player's score is displayed based on how many pipes were successfully passed.

3. Tools and Technologies

- **Programming Language**: Python
- **Game Development Library**: Pygame (Python library for game development)
- **Development Environment**: Any Python IDE or text editor (e.g., VS Code, PyCharm)

4. Game Design and Features

4.1. Game Mechanics

- **Bird Movement**: The bird is affected by gravity, falling downwards unless the player triggers a "flap" action (by pressing the spacebar or clicking the mouse), which causes the bird to rise momentarily.
- **Pipes**: The game generates a series of pipes with gaps that the player must navigate through. The pipes move from right to left, and when they move off-screen, new pipes are created at random intervals.
- **Collisions**: The game detects collisions between the bird and the pipes or the ground. If the bird hits a pipe or the ground, the game ends.
- **Score**: The player’s score increases each time they successfully pass a pipe. The score is displayed at the top of the screen.

- **Game Over**: When a collision occurs, a "Game Over" message is displayed, and the player's final score is shown.

4.2. Visual Design

- **Background**: A scrolling background simulates the motion of the bird through the air.
- **Bird Design**: The bird is represented by a simple image or shape (such as a rectangle) that changes position based on user input.
- **Pipes Design**: The pipes are represented as green vertical columns that span the height of the screen. The gap between pipes is randomly generated.

4.3. Sound Effects

- **Flap Sound**: A sound effect is played when the bird flaps its wings (i.e., when the player presses the spacebar or clicks the mouse).
- **Collision Sound**: A sound effect is played when the bird collides with an obstacle.

5. Implementation

5.1. Python Code Structure

The game is structured using the following components:

- **Main Game Loop**: The core of the game, which continuously updates the game state, processes player input, updates the screen, and handles collisions.
- **Bird Class**: Defines the bird's properties, such as position, speed, and flap behavior. This class also contains methods for updating the bird's position based on gravity and player input.
- **Pipe Class**: Defines the pipes' properties and movement. The pipes are generated at random heights, and they move from right to left on the screen.
- **Collision Detection**: A function checks for collisions between the bird and the pipes or the ground.
- **Game Over Screen**: Displays the final score when the game ends, and offers an option to restart the game.

5.2. Code Example

python
Copy

```
import pygame
import random

# Initialize pygame
pygame.init()

# Set up the display
width = 400
```

```
height = 600
screen = pygame.display.set_mode((width, height))
pygame.display.set_caption("Flappy Bird")

# Colors
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
BLACK = (0, 0, 0)

# Bird properties
bird_width = 40
bird_height = 40
bird_x = 50
bird_y = height // 2
bird_velocity = 0
gravity = 0.5
flap_strength = -10

# Pipe properties
pipe_width = 50
pipe_gap = 150
pipe_velocity = 3

# Game variables
score = 0
clock = pygame.time.Clock()

# Load images
bird_image = pygame.Surface((bird_width, bird_height))
bird_image.fill(BLUE)

# Pipe Class
class Pipe:
    def __init__(self):
        self.x = width
        self.height = random.randint(150, 450)
        self.gap = pipe_gap
        self.y_top = self.height - self.gap // 2
        self.y_bottom = self.height + self.gap // 2
```

```
    def move(self):
        self.x -= pipe_velocity

    def draw(self, screen):
        pygame.draw.rect(screen, GREEN, (self.x, 0, pipe_width,
self.height))  # top pipe
        pygame.draw.rect(screen, GREEN, (self.x, self.y_bottom,
pipe_width, height - self.y_bottom))  # bottom pipe

# Game loop
def game_loop():
    global bird_y, bird_velocity, score

    pipes = [Pipe()]
    running = True
    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                running = False
            if event.type == pygame.KEYDOWN or event.type ==
pygame.MOUSEBUTTONDOWN:
                bird_velocity = flap_strength

        # Update bird position
        bird_velocity += gravity
        bird_y += bird_velocity

        # Prevent bird from falling off the screen
        if bird_y >= height - bird_height:
            bird_y = height - bird_height
            bird_velocity = 0
        if bird_y <= 0:
            bird_y = 0

        # Move and draw pipes
        for pipe in pipes:
            pipe.move()
            pipe.draw(screen)

        # Check for new pipes and collision
        if pipes[-1].x < width - 200:
```

```
            pipes.append(Pipe())

        if pipes[0].x < -pipe_width:
            pipes.pop(0)

        # Check collision with pipes
        for pipe in pipes:
            if bird_x + bird_width > pipe.x and bird_x < pipe.x +
pipe_width:
                if bird_y < pipe.height or bird_y + bird_height >
pipe.y_bottom:
                    running = False

        # Drawing
        screen.fill(WHITE)
        screen.blit(bird_image, (bird_x, bird_y))

        # Update score
        score += 1

        # Display the score
        font = pygame.font.SysFont('Arial', 30)
        score_text = font.render("Score: " + str(score), True,
BLACK)
        screen.blit(score_text, (10, 10))

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

    pygame.quit()

# Run the game
game_loop()
```

5.3. Explanation of Code

- **Game Loop**: The game loop is responsible for handling events (such as player input), updating the game state (bird position, pipes), checking for collisions, and redrawing the screen.
- **Bird Movement**: The bird's vertical position is updated based on gravity. The bird can "flap" upward when the player presses a key or clicks the mouse.

- **Pipes**: Pipes are generated at random heights and move from right to left. The game checks if the bird collides with any pipes and ends the game if so.
- **Score**: The score is incremented each time the bird successfully passes through a pipe and is displayed on the screen.

6. Testing and Results

During testing, the game was checked for:

- Correct collision detection between the bird and pipes.
- Proper rendering of the bird, pipes, and background.
- Accurate scoring system and game over functionality.

The game performed as expected, with the bird successfully navigating the pipes, and the score being updated correctly. The game ended when the bird collided with a pipe or fell to the ground.

7. Conclusion

This project successfully implemented a functional version of the Flappy Bird game using Python and Pygame. The game includes essential features such as player input, collision detection, dynamic obstacles, and a scoring system. The development process helped reinforce key programming concepts such as object-oriented design, game loops, and event handling. Future improvements could include adding more advanced graphics, sound effects, and a high score system.