**Northern University of Business and Technology Khulna**
Department of Computer Science and Engineering

# [Project Report]

Project Title: The Flappy Bird Game

Submitted by:
Muhammad Shabab Sayem
11230321377
sayemshabab@gmail.com

Course Title: Object Oriented programming II (Lab)
Course Code: CSE 2102
Section: 3J

Submitted to:
Shovon Madal [SM]
Lecturer,
Northern University of Business and Technology Khulna

Date of Submission: 22 February 2025

# Contents

# Title:

The Flappy Bird Game


## Abstract:

This project focuses on the development of a classic "Flappy Bird" game using the Java programming language. The project aims to replicate the core gameplay mechanics of the popular mobile game while providing a platform for learning fundamental game development concepts and object-oriented programming principles.

The implementation involves creating game objects such as the bird, pipes, and background, and utilizing Java's Swing library or other suitable frameworks for game rendering and user input. Core game mechanics, including player control, obstacle generation, collision detection, and scoring, are implemented through well-structured and modular code.

This project serves as a valuable learning experience, providing hands-on practice in game design, object-oriented programming, and problem-solving within the context of a familiar and engaging game.


## Introduction:

The Flappy Bird project is a simple 2D game implemented in Java using the Swing and AWT libraries. The game is inspired by the popular Flappy Bird game, where a player controls a bird attempting to navigate through gaps between pipes. The player must press the spacebar to make the bird jump while avoiding collisions with pipes and the ground. The game features a graphical user interface, animated movement, collision detection, and scoring mechanics.

## Literature Review:

[1] The complete documentation of flappy bird game that is desktop-based version of very famous game The Flappy Bird https://www.scribd.com/document/514776142/Flappy-Bird-Documentation.

## Methodology:

(1) Bird: We have to define a class represents that represent the bird character in the game. It will keep track of the bird's position (x, y) and its image.

Fig 1: The bird we will use

(2) Pipe: A pipe class will represent a single pipe obstacle in the game. It keeps track of the pipe's position (x, y), its image, and a flag indicating whether the bird has passed it.

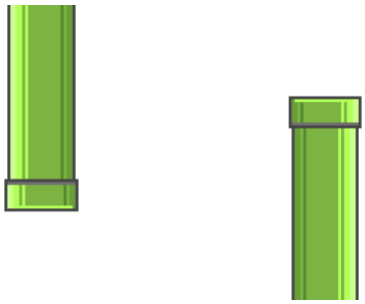

Fig 2: Pipes to be used

(3) Game loop: we need to implement game loop by using a Timer object set to trigger after a small amount of time. In each iteration, the game will call a method to update the positions of the bird and pipes, and then repaints the game canvas.

Fig 3: The background used to draw the pipes and birds on

(4) The move method: This method will update the game state. It will apply gravity to the bird's vertical movement (y-coordinate). It will also move the pipes horizontally to the left (by updating their x-coordinates) to simulate the bird's movement. Additionally, it will check for collisions between the bird and any of the pipes. If a collision is detected, it will be able to end the game.

(5) Placing pipe: There will be a method that will be called periodically by another Timer object to create new pipe obstacles. It will randomly position the top pipe within the upper half of the game canvas and calculate the corresponding bottom pipe position to create a gap for the bird to fly through.

(6) Painting the canvas: There will be a method that will be called whenever the game canvas needs to be repainted. It will draw the background image, the bird character, and all the pipes on the canvas. It will also display the current score or a 'Game Over' message if the game is over.

(7) Key pressing: We need a method that handles the user pressing a key. If the user presses a key, it triggers a jump by setting the bird's vertical velocity to a negative value. Additionally, if the game is over, pressing the key resets the game by restarting the timers and resetting the game variables.

The flow chart for this code:

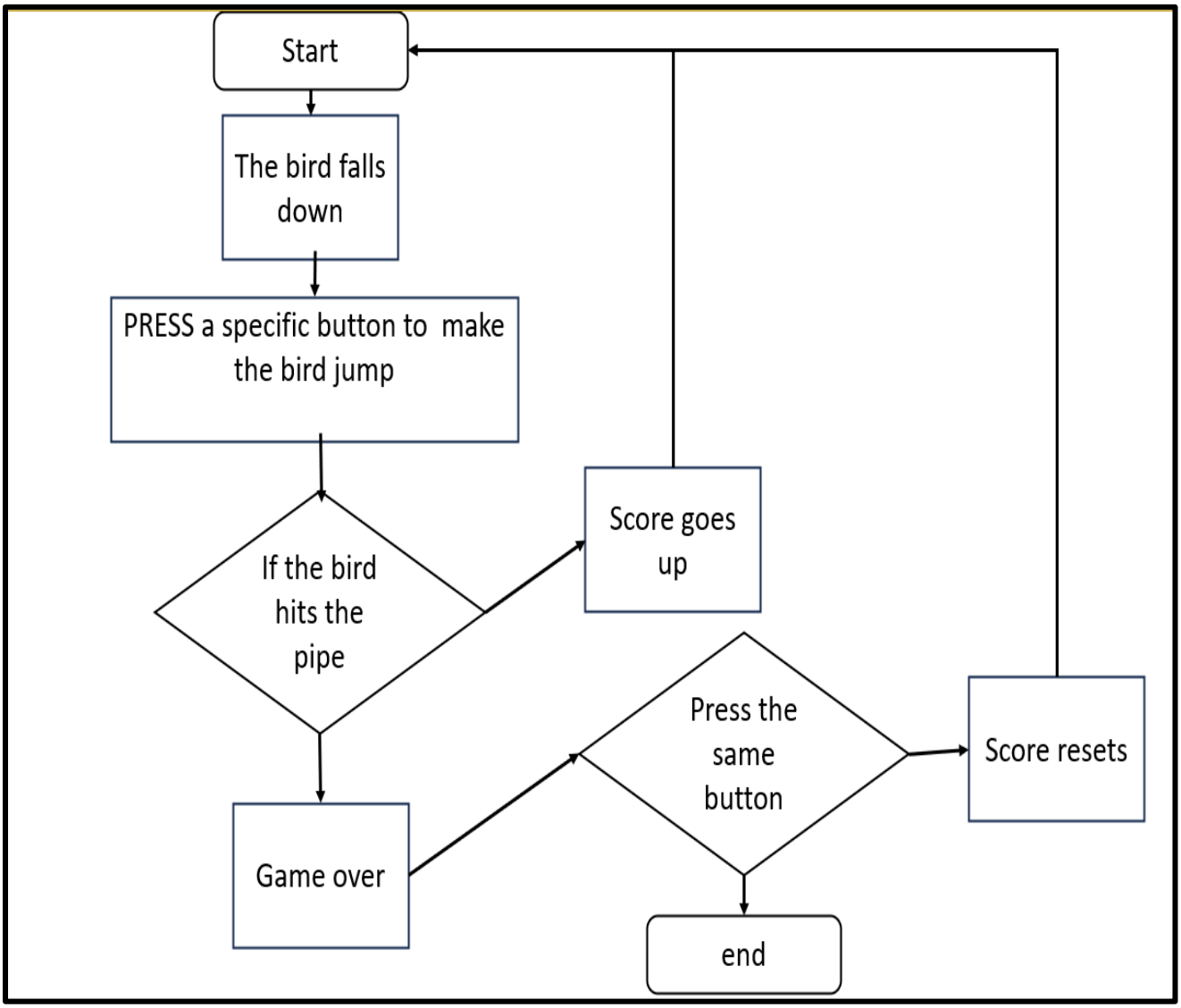


Fig 4: the flowchart for this code

# Implementation:

The Flappy Bird game is designed using Java's AWT and Swing libraries. It follows an event-driven architecture where the game state updates at regular intervals using a Timer, and

user input is handled via keyboard events. The core loop ensures continuous rendering and game physics calculations.

✅ **Game Board (FlappyBird class)**
- Extends JPanel and implements ActionListener and KeyListener
- Handles rendering (paintComponent) and user input (keyPressed)
- Updates game state using a Timer
- Detects collisions and manages game-over conditions

✅ **Bird (Bird class)**
- Represents the player's bird character
- Has properties for position (x, y), size (width, height), and image (img)
- Moves in response to gravity and jumps on spacebar press

✅ **Pipes (Pipe class)**
- Represents obstacles in the game
- Each pipe has a position (x, y), size (width, height), image (img), and a passed flag for scoring
- Moves from right to left at a constant speed

✅ **Game Logic**
- Uses **gravity** and **velocity** to simulate the bird's movement
- Pipes spawn at regular intervals using Timer
- Collision detection ensures the game ends if the bird hits a pipe or falls below the screen

✅ **Timers**
- gameLoop (Timer running at ~60 FPS) updates physics and rendering
- placePipeTimer (Timer running every 1.5s) spawns new pipes

✅ **Rendering**
- Uses Graphics.drawImage() to display the background, bird, and pipes
- Draws the score on the screen
- Stops rendering when the game is over

# Results and Analysis:

## Output:

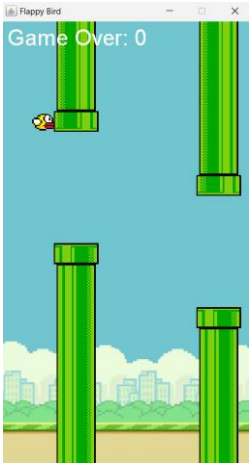


Fig 5: the executed code

## Results:

- **Gameplay Performance**
  - The game runs at **60 FPS**, ensuring a **smooth and responsive** experience.
  - Gravity and jump mechanics work **realistically**, making the gameplay challenging.
- **Collision Detection**
  - The bird **collides accurately** with pipes and the ground.
  - The game correctly **ends** when the bird crashes.
- **Score System**
  - The game keeps track of the **score**, increasing by **1 point** per pipe set passed.
  - The score resets correctly upon restart.
- **Game Flow & Restart**
  - Pressing **SPACEBAR** makes the bird jump.
  - If the game is over, pressing **SPACEBAR** resets the game state.

## Analysis:

- **Simple and efficient** object-oriented design (Bird and Pipe classes).
- Uses **Java Swing Timer** for a **smooth game loop** and **consistent rendering**.
- **Challenging gameplay**, similar to the original Flappy Bird.

# Discussion:

## Areas for Improvement:

🚀 Better Collision Handling: The game immediately stops on collision; adding a small delay or animation could improve user experience.
🚀 Game Restart Enhancement: Instead of restarting instantly, a menu screen could be added.
🚀 Sound Effects & Animations: Adding jump sounds, a game-over effect, and smoother transitions would improve engagement.

## Gameplay Limitations:

❌ Instant Restart Without Delay
- When the game ends, pressing SPACE immediately resets everything.
- Adding a short delay or a "Game Over" screen before restarting would improve the user experience.

❌ Rigid Jump Mechanic
- The bird jumps with a fixed velocity (-9).
- A smoother jump (like an acceleration curve) could make controls more natural.

❌ No Increasing Difficulty

- The pipe movement speed (-4 px/frame) stays constant.
- As the score increases, pipes should move faster to make the game more challenging.

# Conclusion:

**Flappy Bird in Java** is well-structured and functional. It successfully implements core game mechanics such as:

✓ **Gravity and Jump Physics** (bird falls due to gravity and jumps with SPACE)
✓ **Pipe Generation** (pipes spawn at random heights with a gap for the bird to pass)
✓ **Score System** (increments when the bird passes a pipe)
✓ **Collision Detection** (game ends when the bird hits a pipe or the ground)
✓ **Game Loop Using Timers** (handles movement and rendering)
✓ **Restart Mechanism** (resets game state when SPACE is pressed after game over)

# References:

[1] S. S. Santos, "The Science Behind Flappy Bird," 2021 IEEE Integrated STEM Education Conference (ISEC), Princeton, NJ, USA, 2021, pp. 237-237, https://ieeexplore.ieee.org/document/9763893#citations
Doi: 10.1109/ISEC52395.2021.9763893