JAVA (CSL-DC303-A25) ASSIGNMENT REPORT

# Flappy Bird Game

*by*


## Name- Piyush Kumar

## Entry No -23bcs061

## Semester-5th


## Submitted to- Mr. Anuj Mahajan Sir

*Submitted in partial fulfilment of the requirements for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

*in*

**COMPUTER SCIENCE & ENGINEERING**



**SHRI MATA VAISHNO DEVI UNIVERSITY, KATRA**

**(School of Computer Science & Engineering)**

**JAMMU & KASHMIR – 182 320**


**Session 2025-26 (Odd)**

# Description of Project

This is a Java-based recreation of the popular Flappy Bird game. The project uses object-oriented programming principles to handle game logic, rendering, and physics. The player controls a small bird that must navigate through gaps between moving pipes without colliding with them. The game features smooth animations, gravity-based motion, collision detection, and a real-time score counter. It's built using Java's GUI components (such as Swing or AWT) to manage graphics and user interactions through keyboard input.

This project demonstrates foundational concepts in game development, including event handling, frame updates, and simple physics simulation.

**Core game concept**

- The game window is 360x640 pixels, showing a scrolling background, a bird sprite, and pairs of top and bottom pipes that move from right to left.
- The player presses the spacebar to make the bird "jump" upward against gravity and must navigate through the gaps between pipes.

**Main functionalities**

- Bird control and physics: The bird has vertical velocity and is continuously pulled down by gravity; pressing space sets an upward velocity, creating the classic Flappy Bird feel.
- Pipe generation: Pairs of pipes are periodically spawned at random vertical positions with a fixed opening space, simulating an endless obstacle course.
- Collision detection: The game detects overlaps between the bird's rectangle and any pipe rectangle, as well as when the bird falls below the bottom of the board, to trigger game over.
- Scoring system: Each time the bird successfully passes a pair of pipes,

the score increases by 1 (implemented as 0.5 per pipe since there are two pipes per gap).

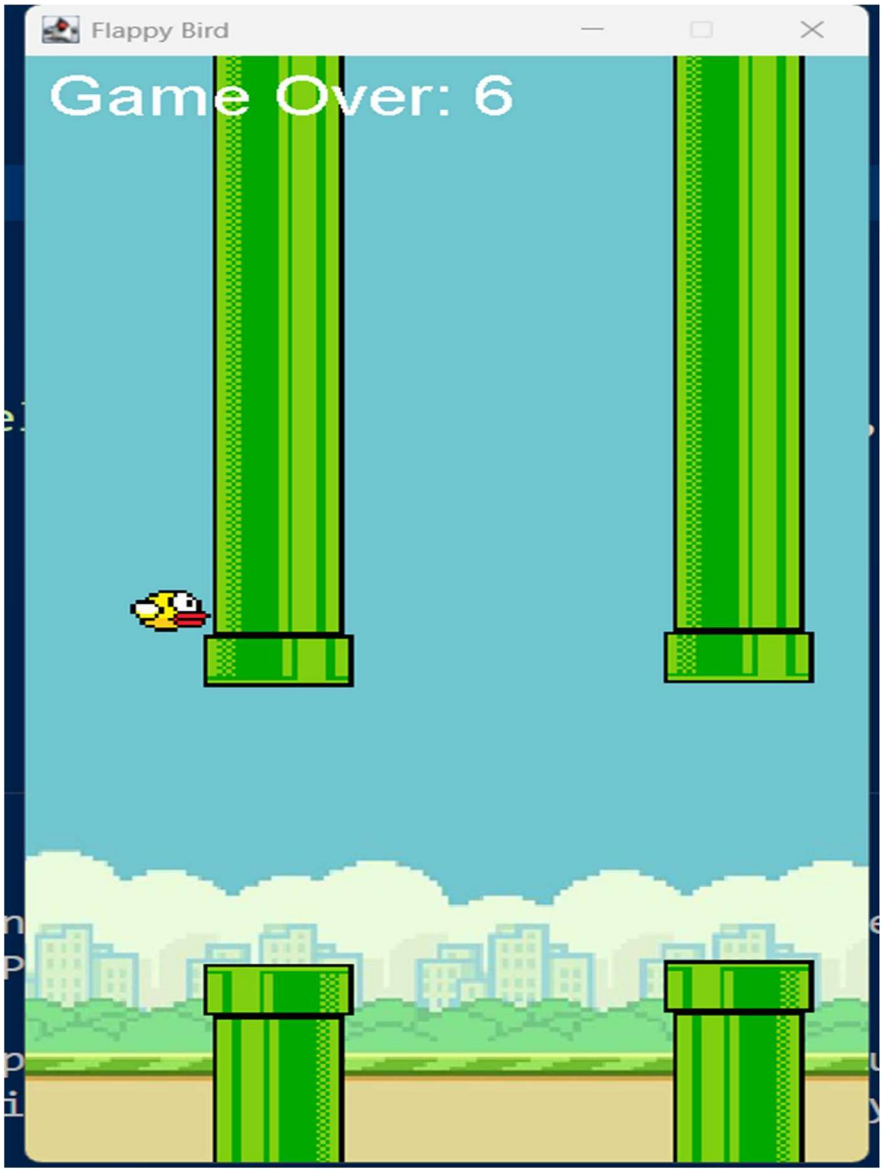
**Game loop and UI**

- Timers are used to implement the game loop (running at 60 FPS) and periodic pipe placement, updating positions and repainting the panel.
- The paintComponent method draws the background, bird, pipes, and current score; on game over, it displays “Game Over: score” at the top.

**Use-cases / user interactions**

- Starting and playing: When the app launches, it opens a fixed-size JFrame titled “Flappy Bird” and immediately starts the game loop and pipe spawning.
- Jumping: Pressing the spacebar makes the bird jump upward to avoid upcoming pipes.
- Restarting: After a game over, pressing space again resets the bird’s position, clears all pipes, resets score and velocities, and restarts both timers for a new run.

# GUI of Project



# Code of flappy bird Game

```
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Random;
import javax.swing.*;

public class FlappyBird extends JPanel implements ActionListener, KeyListener {
    int boardWidth = 360;
    int boardHeight = 640;

    //images
    Image backgroundImg;
    Image birdImg;
    Image topPipeImg;
    Image bottomPipeImg;

    //bird class
    int birdX = boardWidth/8;
    int birdY = boardHeight/2;
    int birdWidth = 34;
    int birdHeight = 24;

    class Bird {
        int x = birdX;
        int y = birdY;
        int width = birdWidth;
        int height = birdHeight;
        Image img;

        Bird(Image img) {
            this.img = img;
        }
    }

    //pipe class
    int pipeX = boardWidth;
    int pipeY = 0;
    int pipeWidth = 64;  //scaled by 1/6
    int pipeHeight = 512;

    class Pipe {
        int x = pipeX;
        int y = pipeY;
        int width = pipeWidth;
        int height = pipeHeight;
        Image img;
        boolean passed = false;

        Pipe(Image img) {
            this.img = img;
        }
    }

    //game logic
```

```
  Bird bird;
  int velocityX = -4; //move pipes to the left speed (simulates bird moving right)
  int velocityY = 0; //move bird up/down speed.
  int gravity = 1;

  ArrayList<Pipe> pipes;
  Random random = new Random();

  Timer gameLoop;
  Timer placePipeTimer;
  boolean gameOver = false;
  double score = 0;

  FlappyBird() {
    setPreferredSize(new Dimension(boardWidth, boardHeight));
    // setBackground(Color.blue);
    setFocusable(true);
    addKeyListener(this);

    //load images
    backgroundImg = new ImageIcon(getClass().getResource("./flappybirdbg.png")).getImage();
    birdImg = new ImageIcon(getClass().getResource("./flappybird.png")).getImage();
    topPipeImg = new ImageIcon(getClass().getResource("./toppipe.png")).getImage();
    bottomPipeImg = new ImageIcon(getClass().getResource("./bottompipe.png")).getImage();

    //bird
    bird = new Bird(birdImg);
    pipes = new ArrayList<Pipe>();

    //place pipes timer
    placePipeTimer = new Timer(1500, new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        // Code to be executed
        placePipes();
      }
    });
    placePipeTimer.start();

    //game timer
    gameLoop = new Timer(1000/60, this); //how long it takes to start timer, milliseconds gone between
frames
    gameLoop.start();
  }

  void placePipes() {
    //(0-1) * pipeHeight/2.
    // 0 -> -128 (pipeHeight/4)
    // 1 -> -128 - 256 (pipeHeight/4 - pipeHeight/2) = -3/4 pipeHeight
    int randomPipeY = (int) (pipeY - pipeHeight/4 - Math.random()*(pipeHeight/2));
    int openingSpace = boardHeight/4;

    Pipe topPipe = new Pipe(topPipeImg);
    topPipe.y = randomPipeY;
    pipes.add(topPipe);

    Pipe bottomPipe = new Pipe(bottomPipeImg);
    bottomPipe.y = topPipe.y  + pipeHeight + openingSpace;
    pipes.add(bottomPipe);
```

```
}


public void paintComponent(Graphics g) {
    super.paintComponent(g);
    draw(g);
}

public void draw(Graphics g) {
    //background
    g.drawImage(backgroundImg, 0, 0, this.boardWidth, this.boardHeight, null);

    //bird
    g.drawImage(birdImg, bird.x, bird.y, bird.width, bird.height, null);

    //pipes
    for (int i = 0; i < pipes.size(); i++) {
        Pipe pipe = pipes.get(i);
        g.drawImage(pipe.img, pipe.x, pipe.y, pipe.width, pipe.height, null);
    }

    //score
    g.setColor(Color.white);

    g.setFont(new Font("Arial", Font.PLAIN, 32));
    if (gameOver) {
        g.drawString("Game Over: " + String.valueOf((int) score), 10, 35);
    }
    else {
        g.drawString(String.valueOf((int) score), 10, 35);
    }

}

public void move() {
    //bird
    velocityY += gravity;
    bird.y += velocityY;
    bird.y = Math.max(bird.y, 0); //apply gravity to current bird.y, limit the bird.y to top of the canvas

    //pipes
    for (int i = 0; i < pipes.size(); i++) {
        Pipe pipe = pipes.get(i);
        pipe.x += velocityX;

        if (!pipe.passed && bird.x > pipe.x + pipe.width) {
            score += 0.5; //0.5 because there are 2 pipes! so 0.5*2 = 1, 1 for each set of pipes
            pipe.passed = true;
        }

        if (collision(bird, pipe)) {
            gameOver = true;
        }
    }

    if (bird.y > boardHeight) {
        gameOver = true;
    }
}
```

```
  boolean collision(Bird a, Pipe b) {
    return a.x < b.x + b.width &&   //a's top left corner doesn't reach b's top right corner
        a.x + a.width > b.x &&   //a's top right corner passes b's top left corner
        a.y < b.y + b.height &&  //a's top left corner doesn't reach b's bottom left corner
        a.y + a.height > b.y;    //a's bottom left corner passes b's top left corner
  }

  @Override
  public void actionPerformed(ActionEvent e) { //called every x milliseconds by gameLoop timer
    move();
    repaint();
    if (gameOver) {
      placePipeTimer.stop();
      gameLoop.stop();
    }
  }

  @Override
  public void keyPressed(KeyEvent e) {
    if (e.getKeyCode() == KeyEvent.VK_SPACE) {
      // System.out.println("JUMP!");
      velocityY = -9;

      if (gameOver) {
        //restart game by resetting conditions
        bird.y = birdY;
        velocityY = 0;
        pipes.clear();
        gameOver = false;
        score = 0;
        gameLoop.start();
        placePipeTimer.start();
      }
    }
  }

  //not needed
  @Override
  public void keyTyped(KeyEvent e) {}

  @Override
  public void keyReleased(KeyEvent e) {}

// ----------------------------------------------------app----------------------------------------------

    public static void main(String[] args)  {
    int boardWidth = 360;
    int boardHeight = 640;

    JFrame frame = new JFrame("Flappy Bird");
    // frame.setVisible(true);
    frame.setSize(boardWidth, boardHeight);
    frame.setLocationRelativeTo(null);
    frame.setResizable(false);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    FlappyBird flappyBird = new FlappyBird();
    frame.add(flappyBird);
```

```
        frame.pack();
        flappyBird.requestFocus();
        frame.setVisible(true);
    }
}
```