**CHAPTER 1**

**INTRODUCTION**

Games have become a fundamental aspect of entertainment and learning in modern computing. Among various genres, 2D arcade-style games continue to captivate users due to their simplicity, challenge, and nostalgic value. One such iconic game is **Flappy Bird**, known for its minimalistic design and addictive gameplay. This project focuses on recreating the Flappy Bird game using Java, utilizing libraries such as **Swing** and **AWT** to design the user interface and implement game logic.

The **Flappy Bird Game in Java** is a graphical application that simulates the original game’s mechanics — a bird flying through gaps between pairs of pipes. The player controls the bird by pressing the spacebar to keep it airborne and navigate through the obstacles without crashing. The objective is to pass through as many pipes as possible to achieve a high score. The project demonstrates animation, collision detection, and real-time input handling using Java’s built-in GUI components.

**Need for a Java-Based Flappy Bird Game**

In an educational and development environment, implementing a game like Flappy Bird in Java offers several key benefits:

**1. Understanding Game Development Fundamentals:**

**Event Handling:** Spacebar input is used to control the bird’s movement.

**Collision Detection:** Helps in understanding object boundary logic and real-time interaction.

**2. Hands-On Practice with GUI and Graphics:**

**Java Swing & AWT Usage:** Utilizes Java’s built-in GUI libraries to create an interactive game window.

**Image Processing:** Demonstrates loading and drawing images such as the background, bird, and pipes.

**3. Logic Building and Problem Solving:**

**Gravity and Physics Simulation:** Bird motion is controlled using a gravity-like effect and jump mechanics.

**Random Pipe Placement:** Pipes appear at random vertical positions to simulate real game unpredictability.

**Scoring System:** Rewards players based on how many pipe sets they pass through.

**4. User Engagement and Replay Value:**

**Simple Controls:** One-button (spacebar) gameplay that is easy to learn but difficult to master.

**Real-Time Feedback:** Score updates and immediate visual feedback for collisions or game-over states.

**Restart Mechanism:** Allows users to restart the game immediately after a crash, enhancing engagement.

**5. Portability:** Pure Java implementation ensures the game can run on any platform with a Java

Virtual Machine (JVM).

**Educational Tool:** Serves as a project for students to learn Java through fun application.

**CHAPTER 2**

**FLAPPY BIRD**

Games have become a fundamental aspect of entertainment and learning in modern computing. Among various genres, 2D arcade-style games continue to captivate users due to their simplicity, challenge, and nostalgic value. One such iconic game is **Flappy Bird**, known for its minimalistic design and addictive gameplay. This project focuses on recreating the Flappy Bird game using Java, utilizing libraries such as **Swing** and **AWT** to design the user interface and implement game Limit Losses: By setting predefined price alerts, users can be notified when a stock price drops below a certain value, allowing them to act quickly and prevent further losses.

**Procedure to create this project:**

* Create a project folder in the C-drive.
* Make sure you place all the java dependencies at there so that we you run the files it will automatically fetch the required dependencies here.
* Create a file named: **FlappyBird.java** (main file where the important things are going to be present)
* Code in FlappyBird.java:

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;

int boardHeight;

Image backgroundImg;

Image birdImg;

Image topPipeImg;

Image bottomPipeImg;

int birdX, birdY, birdWidth, birdHeight;

int pipeWidth, pipeHeight;

int pipeY = 0;

class Bird {

int x, y, width, height;

Image img;

Bird(Image img) {

this.x = birdX;

this.y = birdY;

this.width = birdWidth;

this.height = birdHeight;

this.img = img;

}

}

class Pipe {

int x, y, width, height;

Image img;

boolean passed = false;

Pipe(Image img, int x, int y) {

this.x = x;

this.y = y;

this.width = pipeWidth;

this.height = pipeHeight;

this.img = img;

}

}

Bird bird;

int velocityX = -4;

int velocityY = 0;

int gravity = 1;

ArrayList<Pipe> pipes;

Random random = new Random();

Timer gameLoop;

Timer placePipeTimer;

boolean gameOver = false;

double score = 0;

int lastPipeX = 0;

final int minHorizontalGap = 200;

final int maxExtraGap = 150;

public FlappyBird(int boardWidth, int boardHeight) {

this.boardWidth = boardWidth;

this.boardHeight = boardHeight;

setPreferredSize(new Dimension(boardWidth, boardHeight));

setFocusable(true);

addKeyListener(this);

// Size calculations

birdWidth = boardWidth / 14;

birdHeight = boardHeight / 25;

birdX = boardWidth / 8;

birdY = boardHeight / 2;

pipeWidth = boardWidth / 10;

pipeHeight = boardHeight \* 4 / 5;

// Load images

backgroundImg=new ImageIcon(getClass().getResource("./flappybirdbg.png")).getImage()

.getScaledInstance(boardWidth, boardHeight, Image.SCALE\_SMOOTH);

birdImg = new ImageIcon(getClass().getResource("./flappybird.png")).getImage()

.getScaledInstance(birdWidth, birdHeight, Image.SCALE\_SMOOTH);

topPipeImg = new ImageIcon(getClass().getResource("./toppipe.png")).getImage()

.getScaledInstance(pipeWidth, pipeHeight, Image.SCALE\_SMOOTH);

bottomPipeImg = new ImageIcon(getClass().getResource("./bottompipe.png")).getImage()

.getScaledInstance(pipeWidth, pipeHeight, Image.SCALE\_SMOOTH);

bird = new Bird(birdImg);

pipes = new ArrayList<>();

// Place first pipe immediately

lastPipeX = boardWidth + 100; // off screen

placePipes();

// Start pipe timer

placePipeTimer = new Timer(1000, e -> placePipes());

placePipeTimer.start();

// Game loop

gameLoop = new Timer(1000 / 60, this);

gameLoop.start();

}

void placePipes() {

int openingSpace = boardHeight / 4;

int randomY = (int) (pipeY - pipeHeight / 4 - Math.random() \* (pipeHeight / 2));

int horizontalGap = minHorizontalGap + random.nextInt(maxExtraGap);

int pipeX = lastPipeX + horizontalGap;

Pipe topPipe = new Pipe(topPipeImg, pipeX, randomY);

pipes.add(topPipe);

Pipe bottomPipe = new Pipe(bottomPipeImg, pipeX, randomY + pipeHeight + openingSpace);

pipes.add(bottomPipe);

lastPipeX = pipeX;

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

draw(g);

}

public void draw(Graphics g) {

g.drawImage(backgroundImg, 0, 0, null);

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

for (Pipe pipe : pipes) {

g.drawImage(pipe.img, pipe.x, pipe.y, pipe.width, pipe.height, null);}

g.setColor(Color.white);

g.setFont(new Font("Arial", Font.PLAIN, 32));

if (gameOver) {

g.drawString("Game Over: " + (int) score, 10, 35);

} else {

g.drawString(String.valueOf((int) score), 10, 35);

}

}

public void move() {

velocityY += gravity;

bird.y += velocityY;

bird.y = Math.max(bird.y, 0);

for (Pipe pipe : pipes) {

pipe.x += velocityX;

if (!pipe.passed && bird.x > pipe.x + pipe.width) {

score += 0.5;

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.x + a.width > b.x &&

a.y < b.y + b.height &&

a.y + a.height > b.y;

}

@Override

public void actionPerformed(ActionEvent e) {

move();

repaint();

if (gameOver) {

placePipeTimer.stop();

gameLoop.stop();

}

}

@Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_SPACE) {

velocityY = -9;

if (gameOver) {

bird.y = birdY;

velocityY = 0;

pipes.clear();

lastPipeX = boardWidth + 100;

placePipes();

gameOver = false;

score = 0;

gameLoop.start();

placePipeTimer.start();

}

}

}

@Override public void keyTyped(KeyEvent e) {}

@Override public void keyReleased(KeyEvent e) {}

}

* Create a file named: App.java and its code is below

import javax.swing.\*;

public class App {

public static void main(String[] args) throws Exception {

int boardWidth = 340;

int boardHeight = 460;

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);

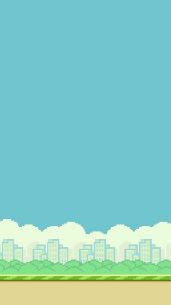
}

}

* Save the following pictures in the same folder:
  + flappybird.png



* + flappybirdbg.png



* + toppipe.png



* + bottompipe.png



* Open the cmd and open the current location where these project files are saved.
* Run below commands:
  + javac FlappyBird.java
  + javac App.java
  + java App

You can see the output: