**Java Code**

package flappyBird;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Rectangle;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.util.ArrayList;

import java.util.Random;

import javax.swing.Timer;

import javax.swing.JFrame;

/\*\*

\*

\* @Muhamaad Syahmi Bin Sairin

\*/

public class FlappyBird implements ActionListener, MouseListener, KeyListener{

public static FlappyBird flappyBird;

public final int WIDTH = 1200, HEIGHT = 900;

public Renderer renderer;

public Rectangle bird;

public ArrayList<Rectangle> columns;

public Random rand;

public int ticks, birdmovement, score, highscore;

public boolean gameover, started;

public FlappyBird(){

JFrame jframe = new JFrame();

Timer timer = new Timer(20,this);

renderer = new Renderer();

rand = new Random();

jframe.addKeyListener(this);

jframe.addMouseListener(this);

jframe.add(renderer);

jframe.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

jframe.setSize(WIDTH, HEIGHT);

jframe.setResizable(false);

jframe.setVisible(true);

jframe.setTitle("Flappy Birdy");

bird = new Rectangle(WIDTH / 2 - 10, HEIGHT / 2 - 10,20 ,20);

columns = new ArrayList<Rectangle>();

addColumn(true);

addColumn(true);

addColumn(true);

addColumn(true);

timer.start();

}

public void addColumn(boolean start){

int space = 300;

int width = 100;

int height = 50 + rand.nextInt(300);

if(start){

columns.add(new Rectangle(WIDTH + width+ columns.size() \* 300, HEIGHT - height - 180, width, height));

columns.add(new Rectangle(WIDTH + width+ (columns.size() -1) \* 300, 0, width, HEIGHT - height - space));

}

else{

columns.add(new Rectangle(columns.get(columns.size() - 1).x + 600, HEIGHT - height - 180, width, height));

columns.add(new Rectangle(columns.get(columns.size() - 1).x, 0, width, HEIGHT - height - space));

}

}

public void paintColumn(Graphics g, Rectangle column){

g.setColor(Color.green.darker().darker());

g.fillRect(column.x, column.y, column.width, column.height);

}

public void jump(){

if(gameover){

bird = new Rectangle(WIDTH / 2 - 10, HEIGHT / 2 - 10,20 ,20);

columns.clear();

birdmovement = 0;

score = 0;

addColumn(true);

addColumn(true);

addColumn(true);

addColumn(true);

gameover = false;

}

if(!started){

started = true;

}

else if(!gameover){

if(birdmovement > 0){

birdmovement = 0;

}

birdmovement -= 10;

}

}

@Override

public void actionPerformed(ActionEvent ae) {

int speed = 10;

ticks++;

if(started){

for(int i=0; i< columns.size(); i++){

Rectangle column = columns.get(i);

column.x -= speed;

}

if(ticks % 2 == 0 && birdmovement < 15){

birdmovement += 2;

}

for(int i=0; i< columns.size(); i++){

Rectangle column = columns.get(i);

if(column.x + column.width < 0){

columns.remove(column);

if(column.y == 0){

addColumn(false);

}

}

}

}

bird.y += birdmovement;

for (Rectangle column : columns){

if (column.y == 0 && bird.x + bird.width / 2 > column.x + column.width / 2 - 10 && bird.x + bird.width / 2 < column.x + column.width / 2 + 10){

score++;

}

if (column.intersects(bird)){

gameover = true;

if (bird.x <= column.x){

bird.x = column.x - bird.width;

}

else{

if (column.y != 0){

bird.y = column.y - bird.height;

}

else if (bird.y < column.height){

bird.y = column.height;

}

}

}

}

if (bird.y > HEIGHT - 180 || bird.y < 0){

gameover = true;

}

if (bird.y + birdmovement >= HEIGHT - 180){

bird.y = HEIGHT - 180 - bird.height;

gameover = true;

}

renderer.repaint();

}

public void repaint(Graphics g) {

g.setColor(Color.cyan);

g.fillRect(0, 0, WIDTH, HEIGHT);

g.setColor(Color.DARK\_GRAY);

g.fillRect(0, HEIGHT - 180, WIDTH, 180);

g.setColor(Color.green);

g.fillRect(0, HEIGHT - 180, WIDTH, 40);

g.setColor(Color.red);

g.fillRect(bird.x, bird.y, bird.width, bird.height);

for(Rectangle column : columns){

paintColumn(g, column);

}

g.setColor(Color.white);

g.setFont(new Font("Tahoma", 1, 90));

if(!started){

g.drawString("Click to play ", 300, HEIGHT / 2 - 50);

}

if(gameover){

g.drawString("Better Luck Next Time.", 85, HEIGHT / 2 - 150);

g.drawString(String.valueOf("Your score is " + score), WIDTH /2 - 350 , 100);

g.drawString("Click to play again", 180, HEIGHT / 2 - 1);

}

if(!gameover && started){

if(score>highscore)

highscore = score;

g.drawString(String.valueOf(score), WIDTH /2 - 50 , 100);

}

}

public static void main(String[]args){

flappyBird = new FlappyBird();

}

@Override

public void mouseClicked(MouseEvent me) {

jump();

}

@Override

public void mousePressed(MouseEvent me) {

}

@Override

public void mouseReleased(MouseEvent me) {

}

@Override

public void mouseEntered(MouseEvent me) {

}

@Override

public void mouseExited(MouseEvent me) {

}

@Override

public void keyTyped(KeyEvent ke) {

}

@Override

public void keyPressed(KeyEvent ke) {

}

@Override

public void keyReleased(KeyEvent ke) {

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

jump();

}

}

}

package flappyBird;

import java.awt.Graphics;

import javax.swing.JPanel;

/\*\*

\*

\* @Muhammad Syahmi Bin Sairin

\*/

public class Renderer extends JPanel{

private static final long serialVersionUID = 1L;

@Override

protected void paintComponent(Graphics g){

super.paintComponent(g);

FlappyBird.flappyBird.repaint(g);

}

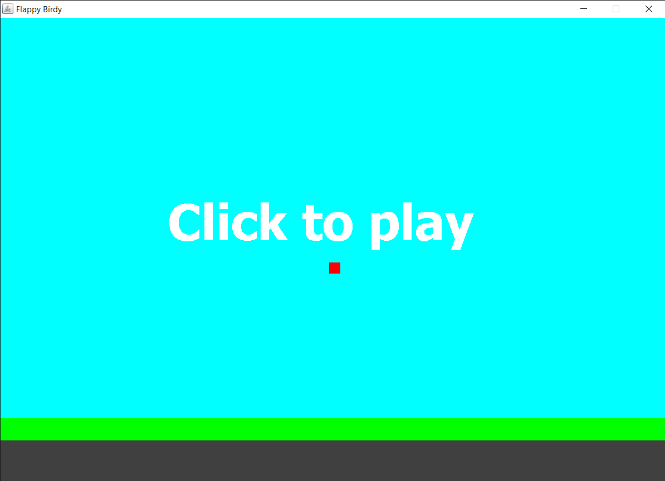
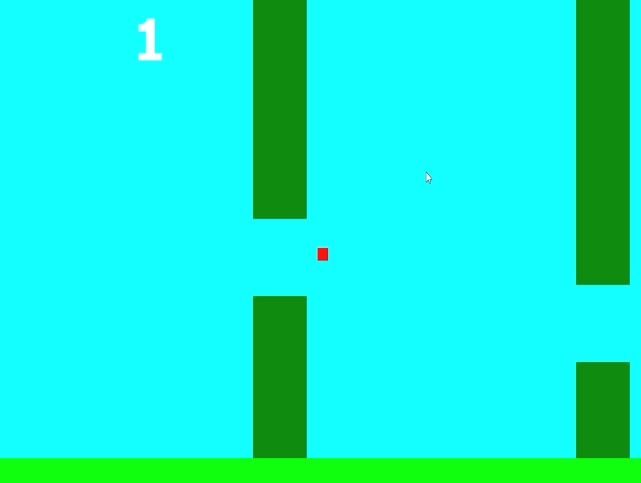
}

**Object Oriented Concept Implementation**

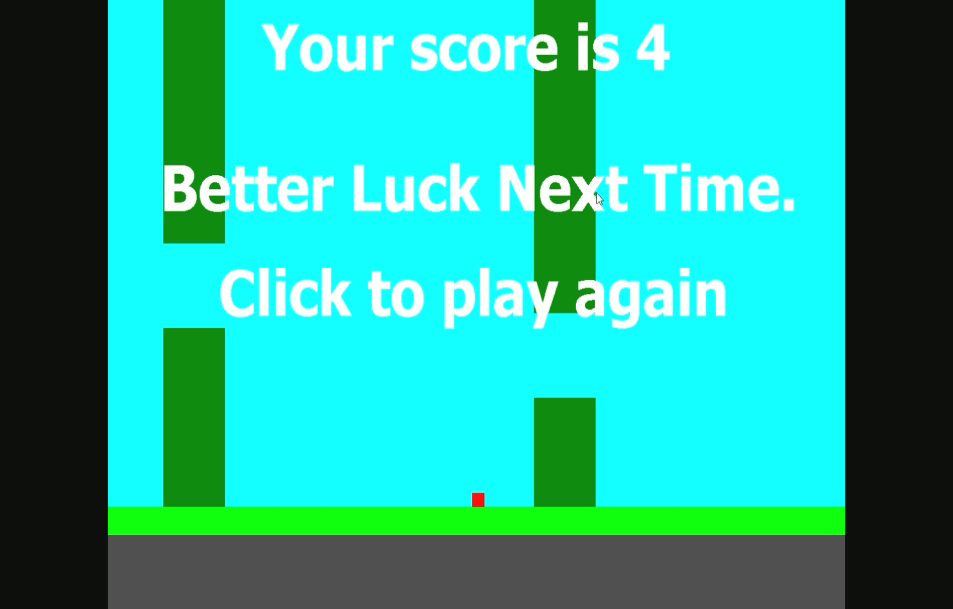
My java project is about a simple game based on Flappy Bird game. User can play the game by clicking user’s mouse to move the box. The objective is to see if user can pass through the wall to collect the highest score. My project applied the five concepts of object-oriented programming (OOP), the first OOP concept would be abstraction since all the processes are hidden from the user and only certain and relevant information are visible. The purpose of abstraction is to hides the underlying complexity of data, presents only the signature of internal functionality and gives flexibility to programmers to change the implementation of the abstract behaviour. The second OOP concept is encapsulation since most of the variables and methods are private so it would not confuse the programmer when calling a variable or method inside a class. Encapsulation is to restricts direct access to data members (fields) of a class, each field has a getter and setter method and setter methods let us change the value of the field. The third one is inheritance, this concept is applied and implements the ActionListener interface. Inheritance is a class (child class) that can extend another class (parent class) by inheriting its features and it improves code reusability. The fourth OOP concept that was applied into the project is polymorphism since each class contains multiple functions because in java polymorphism is refers to the ability to perform a certain action in different ways. It used the same method name several times and different methods of the same name can be called from the object. The fifth OOP concept is association, where all classes are associated with one another whether just one or multiple classes to run the whole programme. Two separate classes are associated through their objects and the two classes are unrelated, each can exist without the other one.

**Read and Write Implementation**

The programme will not run without the inputs that are given to it by the user. As the user interact, the programme will read the inputs that were given by the user and will produce certain action based on what information it received.

** **

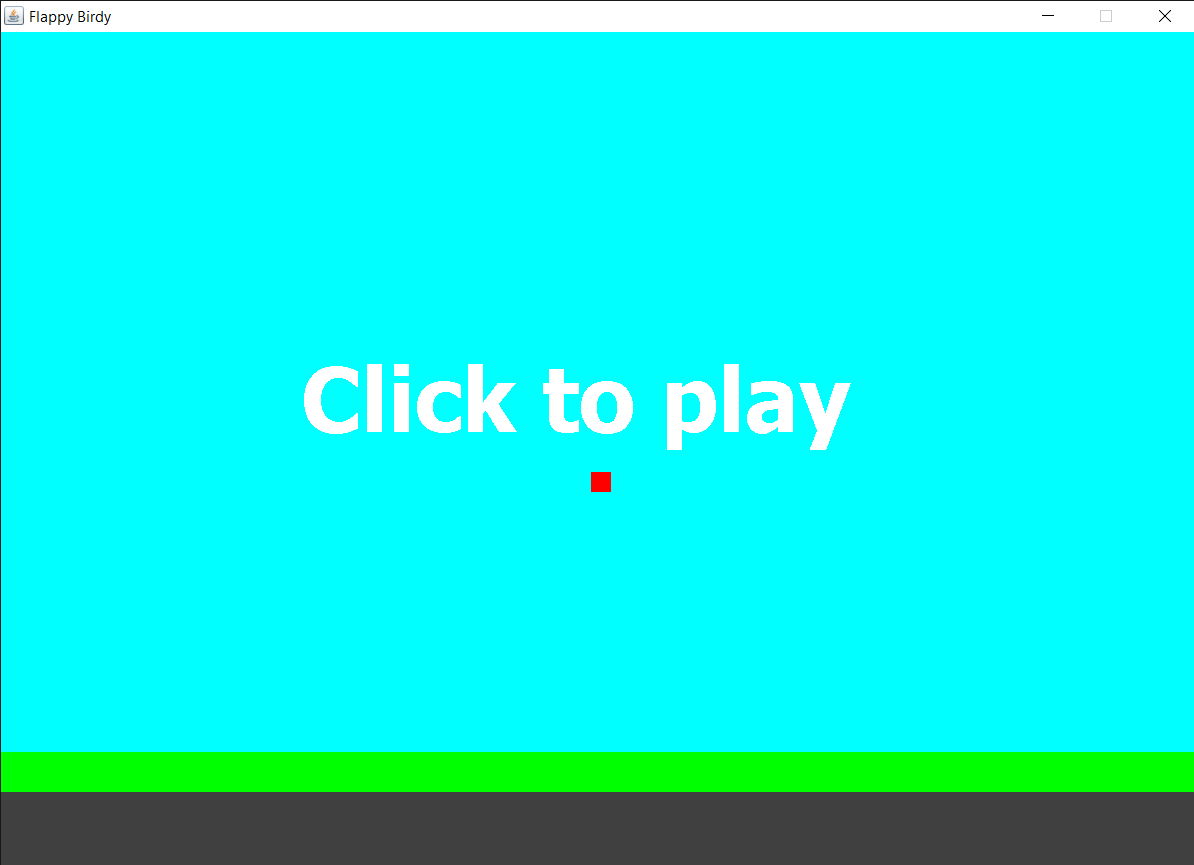
At the beginning of the programme, it will ask the user click their mouse to play the game. The user could only use their mouse to play the game. After the game start, the bird (red rectangle) can move up and down by clicking their left side mouse. The bird will move automatically in a horizontal movement toward the ground and the user will control the bird vertically to pass through the column(pipe). The farther the player play, the more scores the user will achieve. The score will stop if the user failed to pass through the column.



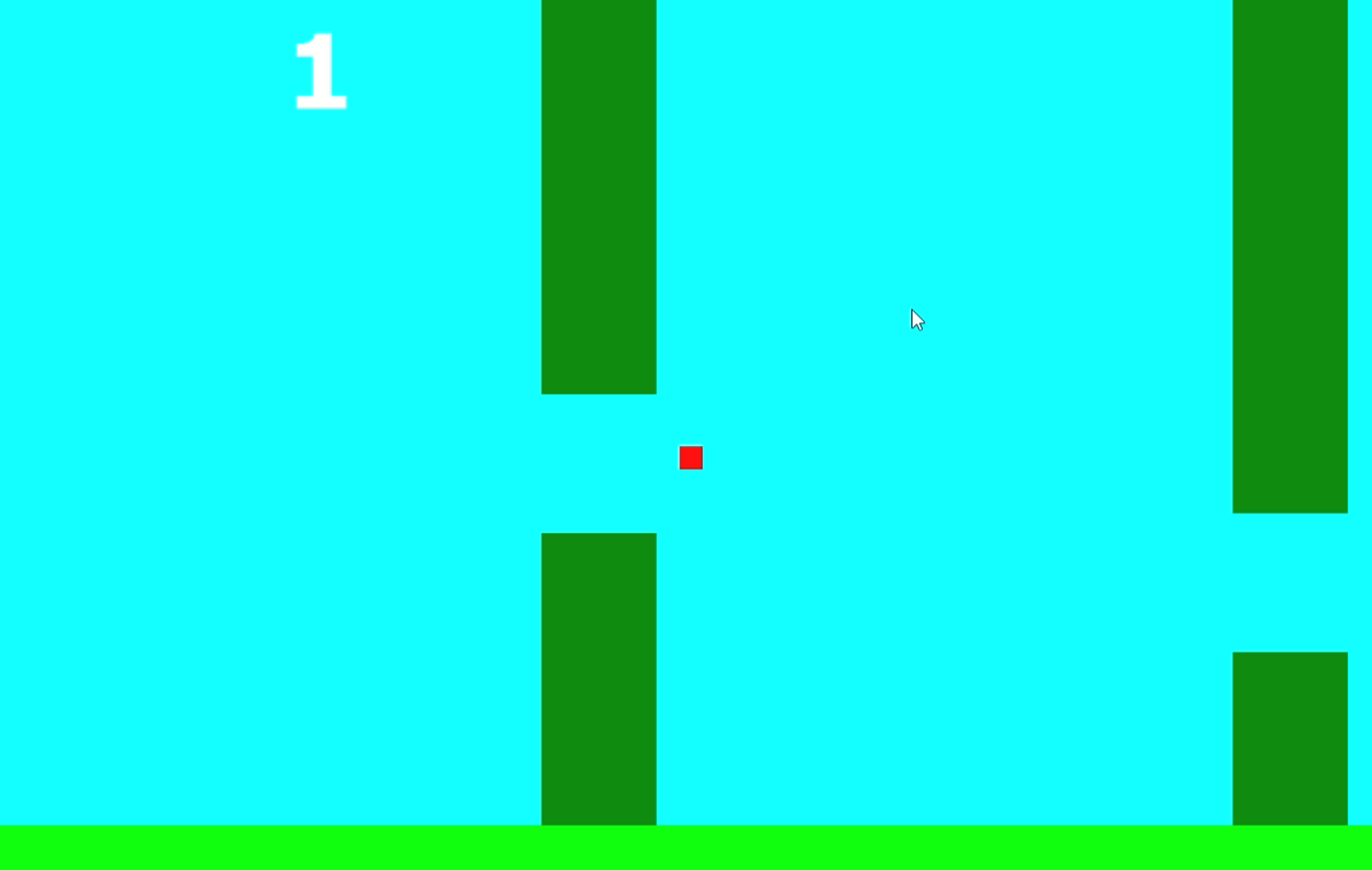
If user touch the ground the game will be over. After the game is over, there will be “Better Luck Next Time” sentence and the user’s score shown on the screen. The user can choose to play again by simply clicking their mouse and the game will start. The game will restart from the beginning and the loop continue until the user exit the programme by clicking the window “X” button.

**User Manual**

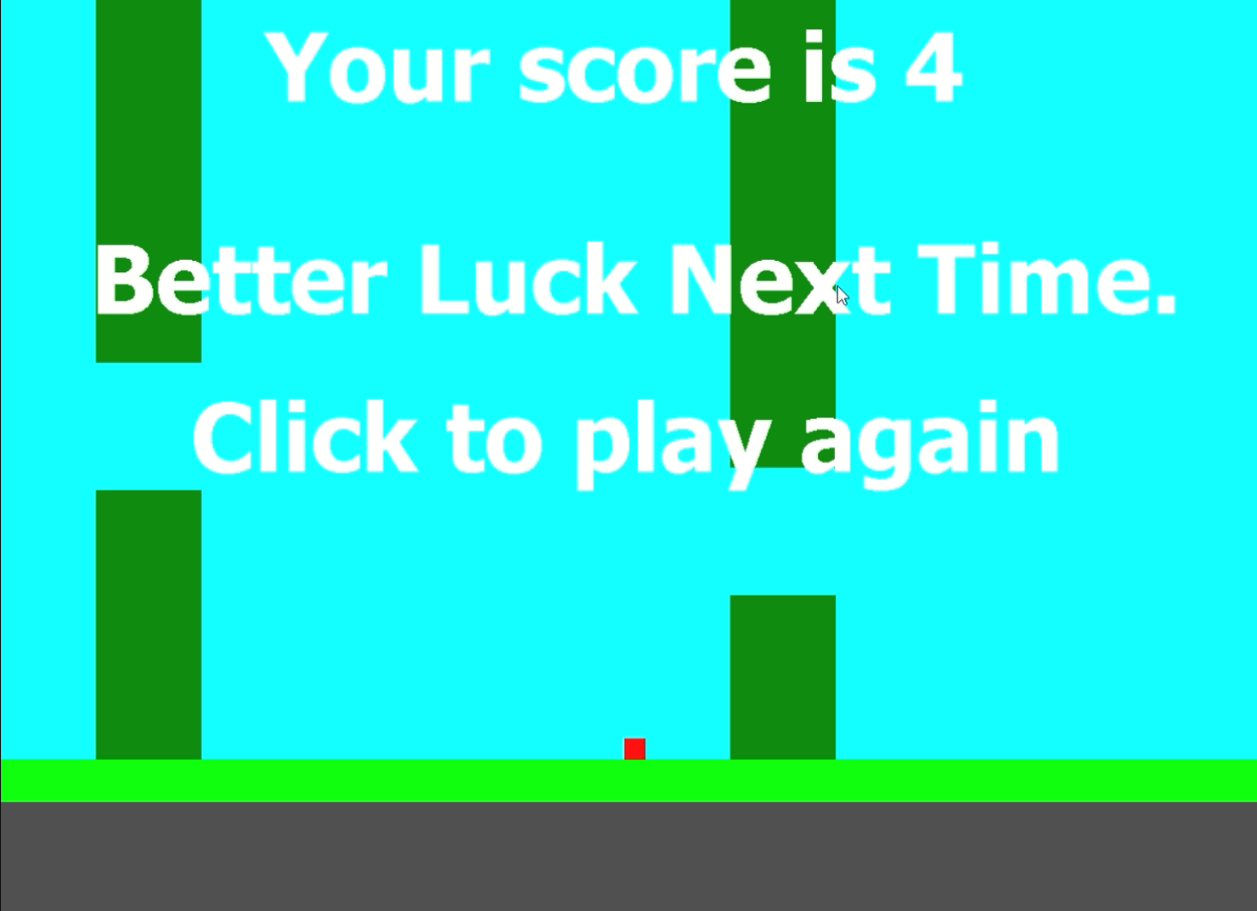
When the user run the programme, a window will appear at the screen and the programme will show:



When the window open, it will ask user “Click to start” to start the game. To start the game the user, need to click their mouse then the game will automatically start. The user could only use their mouse to play the game.



After the game start, the bird (red rectangle) will show randomly on the screen and can be move up and down by the user simply by clicking their mouse. The bird will move automatically in a horizontal movement toward the ground and the user will control the bird vertically to pass through the column(pipe). The farther the bird passes through the column, the higher the user score will be.



If the bird touched the ground, the game will end. When the game is over, the user score will be displayed at the screen and a sentence “Better Luck next Time.” will be shown also. The user can decide whether to continue playing or exit the game. If user want to play the game again, user only need to click their mouse to resume playing and if the user want to exit the game, user can simply click the “X” button on the top right corner of the window and the game will be exited.