**Project on a Snake game created using Java and creating a GUI**

CODE:

package com.mycompany.snakegame;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

public class SnakeGame extends JPanel implements ActionListener {

private static final int BOARD\_WIDTH = 400;

private static final int BOARD\_HEIGHT = 400;

private static final int DOT\_SIZE = 10;

private static final int ALL\_DOTS = (BOARD\_WIDTH \* BOARD\_HEIGHT) / (DOT\_SIZE \* DOT\_SIZE);

private static final int DELAY = 100;

private final int[] x = new int[ALL\_DOTS];

private final int[] y = new int[ALL\_DOTS];

private int dots;

private int foodX;

private int foodY;

private boolean isGameRunning = true;

private boolean isPaused = false;

private boolean isMovingRight = true;

private boolean isMovingLeft = false;

private boolean isMovingUp = false;

private boolean isMovingDown = false;

private Timer timer;

public SnakeGame() {

initGame();

}

private void initGame() {

setBackground(Color.black);

setPreferredSize(new Dimension(BOARD\_WIDTH, BOARD\_HEIGHT));

setFocusable(true);

addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if (key == KeyEvent.VK\_SPACE) {

if (isGameRunning) {

if (isPaused) {

timer.start();

isPaused = false;

} else {

timer.stop();

isPaused = true;

}

}

} else if (key == KeyEvent.VK\_RIGHT && !isMovingLeft) {

isMovingRight = true;

isMovingUp = false;

isMovingDown = false;

} else if (key == KeyEvent.VK\_LEFT && !isMovingRight) {

isMovingLeft = true;

isMovingUp = false;

isMovingDown = false;

} else if (key == KeyEvent.VK\_UP && !isMovingDown) {

isMovingUp = true;

isMovingRight = false;

isMovingLeft = false;

} else if (key == KeyEvent.VK\_DOWN && !isMovingUp) {

isMovingDown = true;

isMovingRight = false;

isMovingLeft = false;

}

}

});

startGame();

}

private void startGame() {

dots = 3; // Initial length of the snake

for (int i = 0; i < dots; i++) {

x[i] = 50 - i \* DOT\_SIZE;

y[i] = 50;

}

placeFood();

timer = new Timer(DELAY, this);

timer.start();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

draw(g);

}

private void draw(Graphics g) {

if (isGameRunning) {

g.setColor(Color.red);

g.fillOval(foodX, foodY, DOT\_SIZE, DOT\_SIZE);

for (int i = 0; i < dots; i++) {

if (i == 0) {

g.setColor(Color.green);

} else {

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

}

g.fillRect(x[i], y[i], DOT\_SIZE, DOT\_SIZE);

}

Toolkit.getDefaultToolkit().sync();

} else {

gameOver(g);

}

}

private void placeFood() {

int r = (int) (Math.random() \* (BOARD\_WIDTH / DOT\_SIZE));

foodX = r \* DOT\_SIZE;

r = (int) (Math.random() \* (BOARD\_HEIGHT / DOT\_SIZE));

foodY = r \* DOT\_SIZE;

}

private void move() {

for (int i = dots; i > 0; i--) {

x[i] = x[i - 1];

y[i] = y[i - 1];

}

if (isMovingRight) {

x[0] += DOT\_SIZE;

} else if (isMovingLeft) {

x[0] -= DOT\_SIZE;

} else if (isMovingUp) {

y[0] -= DOT\_SIZE;

} else if (isMovingDown) {

y[0] += DOT\_SIZE;

}

}

private void checkCollision() {

for (int i = dots; i > 0; i--) {

if ((i > 4) && (x[0] == x[i]) && (y[0] == y[i])) {

isGameRunning = false;

}

}

if (y[0] >= BOARD\_HEIGHT || y[0] < 0 || x[0] >= BOARD\_WIDTH || x[0] < 0) {

isGameRunning = false;

}

if (!isGameRunning) {

timer.stop();

}

}

private void checkFood() {

if ((x[0] == foodX) && (y[0] == foodY)) {

dots++;

placeFood();

}

}

private void gameOver(Graphics g) {

String msg = "Game Over";

Font font = new Font("Arial", Font.BOLD, 30);

FontMetrics metrics = getFontMetrics(font);

g.setColor(Color.white);

g.setFont(font);

g.drawString(msg, (BOARD\_WIDTH - metrics.stringWidth(msg)) / 2, BOARD\_HEIGHT / 2);

String scoreMsg = "Score: " + (dots - 3);

g.drawString(scoreMsg, (BOARD\_WIDTH - metrics.stringWidth(scoreMsg)) / 2, BOARD\_HEIGHT / 2 + 30);

}

@Override

public void actionPerformed(ActionEvent e) {

if (isGameRunning) {

checkFood();

checkCollision();

move();

}

repaint();

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

JFrame frame = new JFrame("Snake Game");

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

frame.setResizable(false);

frame.add(new SnakeGame(), BorderLayout.CENTER);

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

});

}

}

**SCREENSHOTS:**

A computer screen with a black square

Description automatically generated

