/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

package snakegame;

import java.awt.Color;

import javax.swing.BorderFactory;

import javax.swing.JFrame;

/\*\*

\*

\* @author Memon enterprise

\*/

public class GameFrame extends JFrame{

private Splash splash;

private GamePanel gamePanel;

public GameFrame(){

splash = new Splash();

this.add(splash);

this.setTitle("Snake Game");

this.setResizable(false);

this.pack();

this.setLocation(300 , 30);

this.getRootPane().setBorder(BorderFactory.createLineBorder(new Color(47, 79, 47) , 15));

this.setVisible(true);

splash.getStartButton().addActionListener(e -> startGame());

}

/\*\*

\* Starts the game by removing the splash screen and displaying the game panel.

\*/

public void startGame() {

// Remove the splash screen

this.remove(splash);

// Initialize and add the game panel

gamePanel = new GamePanel();

this.add(gamePanel);

// Refresh the frame and adjust its size

this.revalidate();

this.repaint();

this.pack();

// Request focus for the game panel to receive key inputs

gamePanel.requestFocusInWindow();

}

}

package snakegame;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.FontMetrics;

import java.awt.Graphics;

import java.awt.Image;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import javax.swing.JButton;

import javax.swing.JPanel;

import java.util.Random;

import javax.swing.ImageIcon;

import javax.swing.Timer;

/\*\*

\* GamePanel class represents the main game panel for the Snake game.

\*/

public class GamePanel extends JPanel implements ActionListener {

// Constants for game settings

static final int SCREEN\_WIDTH = 600;

static final int SCREEN\_HEIGHT = 600;

static final int UNIT\_SIZE = 25;

static final int GAME\_UNITS = (SCREEN\_WIDTH \* SCREEN\_HEIGHT) / UNIT\_SIZE;

static final int DELAY = 75;

// Arrays to hold the x and y coordinates of the snake's body parts

final int x[] = new int[GAME\_UNITS];

final int y[] = new int[GAME\_UNITS];

// Game state variables

int bodyParts = 6; // Initial number of snake body parts

int applesEaten = 0; // Number of apples eaten

int appleX; // X coordinate of the apple

int appleY; // Y coordinate of the apple

char direction = 'R'; // Initial direction the snake moves ('R' for right)

boolean running = false; // Flag to indicate if the game is running

Timer timer; // Timer to control game speed

Random random; // Random number generator for apple positioning

Image backgroundImage; // Image object for the background

int highestScore = 0; // Variable to track the highest score

JButton playAgainButton; // Button to restart the game

/\*\*

\* Constructor for GamePanel, initializes the game panel settings.

\*/

public GamePanel() {

random = new Random();

this.setPreferredSize(new Dimension(SCREEN\_WIDTH, SCREEN\_HEIGHT));

this.setBackground(Color.BLACK);

this.setFocusable(true);

this.addKeyListener(new MyKeyAdapter());

// Load the background image

backgroundImage = new ImageIcon(getClass().getResource("/Images/snake.jpg")).getImage();

// Initialize the Play Again button

playAgainButton = new JButton("Play Again");

playAgainButton.setFont(new Font("Ink Free", Font.BOLD, 20));

playAgainButton.setFocusPainted(false);

playAgainButton.setVisible(false);

playAgainButton.setBounds((SCREEN\_WIDTH - 150) / 2, SCREEN\_HEIGHT / 2 + 150, 150, 50);

// Customize button appearance

playAgainButton.setOpaque(false); // Make the button transparent

playAgainButton.setContentAreaFilled(false); // Remove the background

playAgainButton.setBorderPainted(false); // Remove the border

playAgainButton.setForeground(Color.WHITE); // Set the text color to white or any other suitable color

playAgainButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

resetGame();

}

});

this.setLayout(null);

this.add(playAgainButton);

startGame();

}

/\*\*

\* Starts the game by placing the first apple and starting the timer.

\*/

public void startGame() {

newApple();

running = true;

timer = new Timer(DELAY, this);

timer.start();

}

/\*\*

\* Resets the game to start over.

\*/

public void resetGame() {

// Reset game state variables

bodyParts = 6;

applesEaten = 0;

direction = 'R';

running = true;

// Reset snake position

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

x[i] = 0;

y[i] = 0;

}

// Hide the Play Again button

playAgainButton.setVisible(false);

// Place a new apple

newApple();

// Restart the timer

if (timer != null) {

timer.stop();

}

timer = new Timer(DELAY, this);

timer.start();

// Repaint the panel to reflect the reset state

repaint();

}

/\*\*

\* Paints the game components on the screen.

\*

\* @param g Graphics object used to draw components

\*/

@Override

public void paintComponent(Graphics g) {

super.paintComponent(g);

draw(g);

}

/\*\*

\* Draws the game elements such as the snake and apple.

\*

\* @param g Graphics object used to draw components

\*/

public void draw(Graphics g) {

if (running) {

// Draw background image

g.drawImage(backgroundImage, 0, 0, this);

// Draw apple

g.setColor(Color.red);

g.fillOval(appleX, appleY, UNIT\_SIZE, UNIT\_SIZE);

// Draw snake

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

if (i == 0) { // Head of the snake

g.setColor(Color.GREEN);

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

} else { // Body of the snake

g.setColor(new Color(45, 180, 0));

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

}

}

// Display current score

g.setColor(Color.red);

g.setFont(new Font("Ink Free", Font.BOLD, 40));

FontMetrics metrics = getFontMetrics(g.getFont());

g.drawString("Score " + applesEaten, (SCREEN\_WIDTH - metrics.stringWidth("Score " + applesEaten)) / 2, g.getFont().getSize());

} else {

// Game over screen

gameOver(g);

}

}

/\*\*

\* Places a new apple on the screen at a random position.

\*/

public void newApple() {

appleX = random.nextInt((int) (SCREEN\_WIDTH / UNIT\_SIZE)) \* UNIT\_SIZE;

appleY = random.nextInt((int) (SCREEN\_HEIGHT / UNIT\_SIZE)) \* UNIT\_SIZE;

}

/\*\*

\* Moves the snake's position based on the current direction.

\*/

public void move() {

// Move body parts

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

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

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

}

// Move head

switch (direction) {

case 'U':

y[0] = y[0] - UNIT\_SIZE;

break;

case 'D':

y[0] = y[0] + UNIT\_SIZE;

break;

case 'L':

x[0] = x[0] - UNIT\_SIZE;

break;

case 'R':

x[0] = x[0] + UNIT\_SIZE;

break;

}

}

/\*\*

\* Checks if the snake has eaten an apple.

\*/

public void checkApple() {

if ((x[0] == appleX) && y[0] == appleY) {

bodyParts++;

applesEaten++;

newApple();

}

}

/\*\*

\* Checks for collisions with the snake's body or the boundaries.

\*/

public void checkCollisions() {

// Check if head collides with body

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

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

running = false;

}

}

// Check if head collides with walls

if (x[0] < 0) {

running = false;

}

if (x[0] >= SCREEN\_WIDTH) {

running = false;

}

if (y[0] < 0) {

running = false;

}

if (y[0] >= SCREEN\_HEIGHT) {

running = false;

}

// Stop the game if not running

if (!running) {

timer.stop();

if (applesEaten > highestScore) {

highestScore = applesEaten;

}

playAgainButton.setVisible(true); // Show the Play Again button

}

}

/\*\*

\* Displays the game over screen with the final score.

\*

\* @param g Graphics object used to draw components

\*/

public void gameOver(Graphics g) {

// Draw background image

g.drawImage(backgroundImage, 0, 0, this);

// Display final score

g.setColor(Color.WHITE);

g.setFont(new Font("Ink Free", Font.BOLD, 40));

FontMetrics metrics1 = getFontMetrics(g.getFont());

g.drawString("Score " + applesEaten, (SCREEN\_WIDTH - metrics1.stringWidth("Score " + applesEaten)) / 2, g.getFont().getSize());

// Display highest score

g.setColor(Color.WHITE);

g.setFont(new Font("Ink Free", Font.BOLD, 40));

g.drawString("Highest Score " + highestScore, ((SCREEN\_WIDTH - metrics1.stringWidth("High Score " + highestScore)) / 2) - 20 , g.getFont().getSize() + 60);

// Display "Game Over" message

g.setColor(Color.WHITE);

g.setFont(new Font("Ink Free", Font.BOLD, 75));

FontMetrics metrics2 = getFontMetrics(g.getFont());

g.drawString("Game Over", (SCREEN\_WIDTH - metrics2.stringWidth("Game Over")) / 2, SCREEN\_HEIGHT / 2);

}

/\*\*

\* Handles the action performed event for the game timer.

\*

\* @param e Action event

\*/

@Override

public void actionPerformed(ActionEvent e) {

if (running) {

move();

checkApple();

checkCollisions();

}

repaint();

}

/\*\*

\* MyKeyAdapter class handles keyboard input for controlling the snake.

\*/

public class MyKeyAdapter extends KeyAdapter {

@Override

public void keyPressed(KeyEvent e) {

switch (e.getKeyCode()) {

case KeyEvent.VK\_LEFT:

if (direction != 'R') {

direction = 'L';

}

break;

case KeyEvent.VK\_RIGHT:

if (direction != 'L') {

direction = 'R';

}

break;

case KeyEvent.VK\_UP:

if (direction != 'D') {

direction = 'U';

}

break;

case KeyEvent.VK\_DOWN:

if (direction != 'U') {

direction = 'D';

}

break;

}

}

}

}

package snakegame;

import java.awt.Color;

import java.awt.Font;

import java.awt.FontMetrics;

import java.awt.Graphics;

import java.awt.Image;

import javax.swing.JButton;

import javax.swing.JPanel;

import javax.swing.BorderFactory;

import javax.swing.ImageIcon;

public class Splash extends JPanel {

private final JButton startButton;

private final Image backgroundImage;

public Splash() {

// Set the layout to null for custom positioning

this.setLayout(null);

// Load the background image

backgroundImage = new ImageIcon(getClass().getResource("/Images/snake.jpg")).getImage();

// Set border

this.setBorder(BorderFactory.createLineBorder(new Color(47, 79, 47), 15));

// Create and configure the Start button

startButton = new JButton("Start Game");

startButton.setFont(new Font("Ink Free", Font.BOLD, 35));

startButton.setBounds(75, 350, 400, 100);

startButton.setFocusPainted(false);

startButton.setContentAreaFilled(false);

startButton.setBorderPainted(false);

startButton.setForeground(Color.WHITE);

startButton.setActionCommand("Start Game");

// Add the button to the panel

this.add(startButton);

this.setPreferredSize(new java.awt.Dimension(600, 600)); // Set size to match game screen

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

drawBackground(g);

drawTitle(g);

}

private void drawBackground(Graphics g) {

// Draw the background image

g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);

}

private void drawTitle(Graphics g) {

// Draw the title

g.setColor(Color.WHITE);

g.setFont(new Font("Ink Free", Font.BOLD, 70));

FontMetrics metrics = getFontMetrics(g.getFont());

String title = "Snake Game";

g.drawString(title, ((getWidth() - metrics.stringWidth(title)) / 2), 250);

}

public JButton getStartButton() {

return startButton;

}

}

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template

\*/

package snakegame;

/\*\*

\*

\* @author Memon enterprise

\*/

public class SnakeGame {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

new GameFrame();

}

}