package brickBracker;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.\*;

class MainMenu {

public static void main(String[] args) {

// Create the main menu window

JFrame frame = new JFrame("Brick Breaker - Select Difficulty");

frame.setSize(700, 600);

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

frame.setLayout(null);

// Title Label

JLabel title = new JLabel("Brick Breaker");

title.setFont(new Font("Serif", Font.BOLD, 40));

title.setForeground(Color.BLUE);

title.setBounds(200, 50, 400, 50);

frame.add(title);

// Difficulty Buttons

JButton easyButton = new JButton("Easy");

JButton mediumButton = new JButton("Medium");

JButton hardButton = new JButton("Hard");

easyButton.setBounds(250, 150, 200, 50);

mediumButton.setBounds(250, 250, 200, 50);

hardButton.setBounds(250, 350, 200, 50);

// Add action listeners to buttons

easyButton.addActionListener(e -> startGame("Easy", frame));

mediumButton.addActionListener(e -> startGame("Medium", frame));

hardButton.addActionListener(e -> startGame("Hard", frame));

// Add buttons to frame

frame.add(easyButton);

frame.add(mediumButton);

frame.add(hardButton);

// Show the frame

frame.setVisible(true);

}

private static void startGame(String difficulty, JFrame frame) {

// Close the menu frame

frame.dispose();

// Create a new JFrame for the game

JFrame gameFrame = new JFrame("Brick Breaker - " + difficulty + " Level");

Gameplay gameplay = new Gameplay();

gameplay.setDifficulty(difficulty);

gameFrame.setBounds(10, 10, 700, 600);

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

gameFrame.setResizable(false);

gameFrame.add(gameplay);

gameFrame.setVisible(true);

    }

}

package brickBracker;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import javax.swing.\*;

public class Gameplay extends JPanel implements KeyListener, ActionListener {

private boolean play = false;

private int score = 0;

private int totalBricks = 21;

private Timer timer;

private int delay = 8; // Default delay for Easy level

private int playerX = 310;

private int ballposX = 120;

private int ballposY = 350;

private int ballXdir = -1; // Default ball X direction

private int ballYdir = -2; // Default ball Y direction

private String difficultyLevel = "Easy"; // Default difficulty

private MapGenerator map;

public Gameplay() {

map = new MapGenerator(3, 7);

addKeyListener(this);

setFocusable(true);

setFocusTraversalKeysEnabled(false);

timer = new Timer(delay, this); // Timer with default delay

timer.start();

}

public void setDifficulty(String level) {

difficultyLevel = level;

switch (difficultyLevel) {

case "Medium":

delay = 6; // Medium speed

ballXdir = -2;

ballYdir = -3;

break;

case "Hard":

delay = 4; // Hard speed

ballXdir = -3;

ballYdir = -4;

break;

default: // Easy

delay = 8; // Easy speed

ballXdir = -1;

ballYdir = -2;

break;

}

if (timer != null) {

timer.setDelay(delay); // Update the timer with the new delay

}

}

public void paint(Graphics g) {

// Background

Graphics2D g2d = (Graphics2D) g;

GradientPaint gradient = new GradientPaint(0, 0, Color.CYAN, 0, getHeight(), new Color(25, 37, 126));

g2d.setPaint(gradient);

g2d.fillRect(1, 1, 692, 592);

// Drawing map

map.draw((Graphics2D) g);

g.setColor(Color.LIGHT\_GRAY);

g.fillRect(0, 0, 3, 592);

g.fillRect(0, 0, 692, 3);

g.fillRect(691, 0, 3, 592);

// Scores

g.setColor(Color.WHITE);

g.setFont(new Font("serif", Font.BOLD, 25));

g.drawString("Score: " + score, 540, 30);

// Difficulty Level Display

g.setFont(new Font("serif", Font.BOLD, 20));

g.drawString("Difficulty: " + difficultyLevel, 20, 30);

// Paddle

g.setColor(Color.GREEN);

g.fillRect(playerX, 550, 100, 8);

// Ball

g2d.setColor(new Color(255, 165, 0, 128)); // Semi-transparent orange for glow

g2d.fillOval(ballposX - 5, ballposY - 5, 30, 30); // Glow radius

g.setColor(Color.ORANGE);

g.fillOval(ballposX, ballposY, 20, 20);

// Win Condition

if (totalBricks <= 0) {

play = false;

ballXdir = 0;

ballYdir = 0;

g.setColor(new Color(255, 215, 0)); // Golden Yellow for "You Won!"

g.setFont(new Font("serif", Font.BOLD, 30));

g.drawString("You Won! Score: " + score, 190, 300);

g.setFont(new Font("serif", Font.BOLD, 20));

g.drawString("Press Enter to Restart", 230, 350);

}

// Game Over scenario

if (ballposY > 570) {

play = false;

ballXdir = 0;

ballYdir = 0;

g.setColor(new Color(255, 255, 255)); // Bright White for "Game Over!"

g.setFont(new Font("serif", Font.BOLD, 30));

g.drawString("Game Over! Score: " + score, 190, 300);

g.setFont(new Font("serif", Font.BOLD, 20));

g.drawString("Press Enter to Restart", 230, 350);

}

g.dispose();

}

@Override

public void actionPerformed(ActionEvent e) {

timer.start();

if (play) {

if (new Rectangle(ballposX, ballposY, 20, 20).intersects(new Rectangle(playerX, 550, 100, 8))) {

ballYdir = -ballYdir;

}

// Ball Collision with Bricks

A: for (int i = 0; i < map.map.length; i++) {

for (int j = 0; j < map.map[0].length; j++) {

if (map.map[i][j] > 0) {

int brickX = j \* map.brickWidth + 80;

int brickY = i \* map.brickHeight + 80;

int brickWidth = map.brickWidth;

int brickHeight = map.brickHeight;

Rectangle rect = new Rectangle(brickX, brickY, brickWidth, brickHeight);

Rectangle ballRect = new Rectangle(ballposX, ballposY, 20, 20);

if (ballRect.intersects(rect)) {

map.setBrickValue(0, i, j);

totalBricks--;

score += 5;

if (ballposX + 19 <= rect.x || ballposX + 1 >= rect.x + rect.width) {

ballXdir = -ballXdir;

} else {

ballYdir = -ballYdir;

}

break A;

}

}

}

}

// Ball Movement

ballposX += ballXdir;

ballposY += ballYdir;

// Ball Collision with Walls

if (ballposX < 0 || ballposX > 670) ballXdir = -ballXdir;

if (ballposY < 0) ballYdir = -ballYdir;

}

repaint();

}

@Override

public void keyPressed(KeyEvent e) {

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

if (playerX >= 600) {

playerX = 600;

} else {

moveRight();

}

}

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

if (playerX <= 10) {

playerX = 10;

} else {

moveLeft();

}

}

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

if (!play) {

play = true;

ballposX = 120;

ballposY = 350;

playerX = 310;

score = 0;

totalBricks = 21;

map = new MapGenerator(3, 7);

// Reset difficulty settings

setDifficulty(difficultyLevel);

repaint();

}

}

}

public void moveRight() {

play = true;

playerX += 20;

}

public void moveLeft() {

play = true;

playerX -= 20;

}

@Override

public void keyTyped(KeyEvent e) {}

@Override

public void keyReleased(KeyEvent e) {}

}

package brickBracker;

import java.awt.\*;

public class MapGenerator {

public int map[][];

public int brickWidth;

public int brickHeight;

public MapGenerator(int row, int col) {

map = new int[row][col];

for (int i = 0; i < map.length; i++) {

for (int j = 0; j < map[0].length; j++) {

map[i][j] = 1; // 1 means the brick exists

}

}

brickWidth = 540 / col;

brickHeight = 150 / row;

}

public void draw(Graphics2D g) {

// Define muted, professional colors for each row

Color[] rowColors = {

new Color(44, 62, 80), // Dark Grayish Blue

new Color(52, 152, 219), // Soft Blue

new Color(46, 204, 113), // Soft Green

new Color(241, 196, 15), // Muted Yellow

new Color(231, 76, 60), // Soft Red

};

for (int i = 0; i < map.length; i++) {

for (int j = 0; j < map[0].length; j++) {

if (map[i][j] > 0) {

// Use alternating professional colors

g.setColor(rowColors[i % rowColors.length]);

g.fillRect(j \* brickWidth + 88, i \* brickHeight + 50, brickWidth, brickHeight);

// Brick border for definition

g.setStroke(new BasicStroke(2));

g.setColor(Color.BLACK);

g.drawRect(j \* brickWidth + 88, i \* brickHeight + 50, brickWidth, brickHeight);

}

}

}

}

public void setBrickValue(int value, int row, int col) {

map[row][col] = value; // Set brick value (0 = broken, 1 = exists)

    }

}