Project 5

**Name of the project:** **Brick Breaker Game** **Date:28/07/24**

**Name: Hasanapuram Anil kumar**

**Project Overivew:**

The Brick Breaker game in Java is a classic arcade-style game where the player controls a paddle to bounce a ball and break a wall of bricks. The objective is to clear all the bricks using the ball without letting it fall below the paddle. This project is an excellent introduction to game development and Java programming, offering a hands-on experience with graphics, user input, and game logic.

**Key Features to Add:**

* Graphics and GUI: Implement a visual interface with colorful bricks, a paddle, and a bouncing ball using Java's graphics libraries.
* Game Logic: Develop core mechanics for paddle movement, ball bouncing, and brick destruction. Include collision detection and a scoring system.
* Levels and Difficulty: Design multiple levels with increasing complexity and adjust difficulty by altering ball speed and introducing challenges.
* User Input: Enable keyboard input for paddle control, and add features like pausing and restarting.
* Sound Effects and Music: Enhance the gaming experience with sound effects for ball hits, brick breaks, and background music.
* Game Over and Victory Conditions: Implement conditions for game over and define victory conditions for completing levels.
* Optimizations: Optimize code for better performance, considering features like double buffering.
* Documentation: Provide clear documentation and comments for the codebase to aid understanding.

**INPUT:**

package oops;

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;

import java.awt.event.KeyListener;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

public class BrickBreakerGame extends JFrame {

private static final int WIDTH = 800;

private static final int HEIGHT = 600;

private static final int PADDLE\_WIDTH = 100;

private static final int PADDLE\_HEIGHT = 20;

private static final int BALL\_SIZE = 20;

private static final int BRICK\_WIDTH = 75;

private static final int BRICK\_HEIGHT = 30;

private static final int NUM\_BRICKS\_X = 10;

private static final int NUM\_BRICKS\_Y = 5;

private Paddle paddle;

private Ball ball;

private List<Brick> bricks;

private Timer timer;

private int score = 0;

public BrickBreakerGame() {

setTitle("Brick Breaker");

setSize(WIDTH, HEIGHT);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setResizable(false);

paddle = new Paddle(WIDTH / 2 - PADDLE\_WIDTH / 2, HEIGHT - PADDLE\_HEIGHT - 30);

ball = new Ball(WIDTH / 2 - BALL\_SIZE / 2, HEIGHT - PADDLE\_HEIGHT - BALL\_SIZE - 40);

bricks = new ArrayList<>();

for (int i = 0; i < NUM\_BRICKS\_X; i++) {

for (int j = 0; j < NUM\_BRICKS\_Y; j++) {

bricks.add(new Brick(i \* (BRICK\_WIDTH + 10) + 30, j \* (BRICK\_HEIGHT + 10) + 50));

}

}

timer = new Timer(10, new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

ball.move();

checkCollisions();

repaint();

}

});

timer.start();

addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) {

paddle.keyPressed(e);

}

@Override

public void keyReleased(KeyEvent e) {

paddle.keyReleased(e);

}

});

setFocusable(true);

add(new GamePanel());

}

private void checkCollisions() {

// Paddle collision

if (ball.getBounds().intersects(paddle.getBounds())) {

ball.reverseYDirection();

}

// Brick collision

for (Brick brick : bricks) {

if (ball.getBounds().intersects(brick.getBounds())) {

ball.reverseYDirection();

bricks.remove(brick);

score++;

break;

}

}

// Ball out of bounds

if (ball.getY() > HEIGHT) {

timer.stop();

JOptionPane.showMessageDialog(this, "Game Over! Your Score: " + score);

System.exit(0);

}

}

private class GamePanel extends JPanel {

public GamePanel() {

setPreferredSize(new Dimension(WIDTH, HEIGHT));

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

paddle.draw(g);

ball.draw(g);

for (Brick brick : bricks) {

brick.draw(g);

}

}

}

private class Paddle {

private int x, y;

private int dx = 0;

public Paddle(int x, int y) {

this.x = x;

this.y = y;

}

public void draw(Graphics g) {

g.setColor(Color.BLACK);

g.fillRect(x, y, PADDLE\_WIDTH, PADDLE\_HEIGHT);

}

public void move() {

x += dx;

if (x < 0) x = 0;

if (x > WIDTH - PADDLE\_WIDTH) x = WIDTH - PADDLE\_WIDTH;

}

public void keyPressed(KeyEvent e) {

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

dx = -5;

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

dx = 5;

}

}

public void keyReleased(KeyEvent e) {

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

dx = 0;

}

}

public Rectangle getBounds() {

return new Rectangle(x, y, PADDLE\_WIDTH, PADDLE\_HEIGHT);

}

}

private class Ball {

private int x, y;

private int dx = 2, dy = -2;

public Ball(int x, int y) {

this.x = x;

this.y = y;

}

public int getY()

{

// TODO Auto-generated method stub

return 0;

}

public void draw(Graphics g) {

g.setColor(Color.RED);

g.fillOval(x, y, BALL\_SIZE, BALL\_SIZE);

}

public void move() {

x += dx;

y += dy;

if (x < 0 || x > WIDTH - BALL\_SIZE) dx = -dx;

if (y < 0) dy = -dy;

}

public void reverseYDirection() {

dy = -dy;

}

public Rectangle getBounds() {

return new Rectangle(x, y, BALL\_SIZE, BALL\_SIZE);

}

}

private class Brick {

private int x, y;

public Brick(int x, int y) {

this.x = x;

this.y = y;

}

public void draw(Graphics g) {

g.setColor(Color.BLUE);

g.fillRect(x, y, BRICK\_WIDTH, BRICK\_HEIGHT);

}

public Rectangle getBounds() {

return new Rectangle(x, y, BRICK\_WIDTH, BRICK\_HEIGHT);

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> new BrickBreakerGame().setVisible(true));

}

}

**0UTPUT:**

