A Project Report

Submitted in partial Fulfillment of the Requirement for the Award of the Degree of

BACHELOR OF TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

TO

DR.AP.J. ABDUL KALAM TECHNICAL UNIVERSITY

LUCKNOW

SUBMITTED BY

PRIYANSHU SINGH (2300101530107)

SAMRTH BISHT (2300101530128)

SHRISHTI SONKAR (2300101530153)

KARTIKEY JAISWAL (2300101530069)

UNDER THE SUPERVISION OF

MR. AJAI KUMAR MAURYA

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UNITED COLLEGE OF ENGINEERING AND RESEARCH, PRAYAGRAJ

CANDIDATE’S DECLARATION

We **PRIYANSHU SINGH**, **SAMRTH BISHT**, **SHRISHTI SONKAR and KARTIKEY JAISWAL,** students of Bachelor of Technology of Computer Science and Engineering hereby declared that we own the full responsibility for the information, results etc provided in this mini project titled **“BRICK BREAKER”** submitted to Dr. A.P.J Abdul Kalam Technical University, Lucknow for award of Bachelor of Technology (Computer Science and Engineering) degree. We have taken care in all respect to honor the intellectual property right and have acknowledged the contributions of others for using them in this academic purpose. We further declared that in case of any violation of intellectual property right or copyright, we as the candidate would be fully responsible for the same. Our supervisor and institute should not be held for full or partial violation of copy right if found at any stage of our degree.

Date: **PRIYANSHU SINGH (2300101530107)**

Place: Prayagraj **SAMARTH BISHT (23001015300128)**

SHRISHTI SONKAR (2300101530153)

KARTIKEY JAISWAL (2300101530069)

CERTIFICATE

This is to certify that the mini project work entitled **"BRICK BREAKER",** submitted by PRIYANSHU SINGH, SAMARTH BISHT, SHRISHTI SONKAR and KARTIKEY JAISWAL to the Dr. A.P.J. Abdul Kalam Technical University, Lucknow, for the partial fulfillment of the requirement for the award of **Bachelor of Technology (Computer Science and Engineering)** degree, is a record of student's own study carried out under my supervision and guidance.

This mini project has not been submitted to any other university or institution for the award of any other degree.

SUPERVISOR

(MR. AJAI KUMAR MAURYA)

TABLE OF CONTENT

Introduction

Title: Brick Breaker: The Arcade Game

Here is a well-structured summary for Welcome to "Brick Breaker" - an exciting game developed in Java. This classic arcade-style game challenges you to break bricks using a ball and paddle. Strategize your moves, aim precisely, and break all the bricks to advance to the next level. Enjoy the simple yet addictive gameplay and see how high you can score. Embark on this fun-filled journey and hone your gaming skills with "Brick Breaker"!

Problem Statement

Design an engaging brick breaker game using Java programming. The game involves controlling a paddle to bounce a ball and break bricks positioned at the top of the screen. Players must aim to break all the bricks while preventing the ball from falling off the bottom of the screen. The game should provide feedback on the number of lives left, the player's score, and the number of bricks remaining. Ensure that the game offers increasing difficulty levels and maintains user engagement through visual and sound effects.

Proposed Model

The model that I would wish to propose here is designing a brick breaker game in Java, which shall include the following features: block arrangement at the top of the screen, a movable paddle at the bottom controlled by the user, and a ball that bounces off the paddle and blocks.

• **Gameplay Mechanics: The player uses the paddle to keep the ball in play while aiming to break all the bricks.**

• **Scoring Mechanism: The player earns points for each brick broken. Scores can be increased with each subsequent level of difficulty.**

• **Game Loop: The game continues until the player loses all their lives by failing to keep the ball in play. Feedback and scoring update according to performance in the loop.**

****1. DFDs (Data Flow Diagrams) ****

****– Zero Level DFD (CLD)**

In the zero-level Data Flow Diagram, known as the Context Level Diagram, the main process is "Brick Breaker Game." The

external entity is the User, and the system interacts with the user through input (paddle movements) and output (feedback and score).

****– Level-1 Data Flow Diagram****

At Level-1, the DFD breaks down the main process into sub-processes:

1. **Start Game: It initializes the game by setting up the paddle, ball, and brick layout.**

2. **Move Paddle: Listens for the user's input to move the paddle left or right.**

3. **Ball Movement: Calculates and updates the ball's position based on collisions with the walls, paddle, and bricks.**

4. **Check Collision: Detects collisions between the ball and bricks, updating the brick status and user score accordingly.**

5. **Update Score: Alters the user's score based on the number of bricks broken.**

6. **Game Over: Displays the final score and ends the game if the player loses all their lives or clears all the bricks.**

****2. ERD (Entity Relationship Diagram) ****

Being a very simple game, the ERD will be quite simple too:

- **Entities:

- **User**: This represents who is playing the game.

- **Game Session**: This would be for each different session of a game that stores the current score, the number of lives left, and the brick layout.

Source code:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class BrickBreaker extends JPanel implements KeyListener,
ActionListener {
private int ballX = 200;
private int ballY = 300;
private int ballXDir = -1;
private int ballYDir = -2;
private int playerX = 310;
private boolean play = false;
private int score = 0;
private int totalBricks = 21;
private Timer timer;
private int delay = 8;
private int[][] bricks;

public BrickBreaker() {
  bricks = new int[3][7];
```

```
for (int i = 0; i < bricks.length; i++) {
for (int j = 0; j < bricks[0].length; j++) {
bricks[i][j] = 1;
}
}

addKeyListener(this);
setFocusable(true);
setFocusTraversalKeysEnabled(false);
timer = new Timer(delay, this);
timer.start();
}

public void paint(Graphics g) {
// Background
g.setColor(Color.black);
g.fillRect(1, 1, 692, 592);

// Borders
g.setColor(Color.yellow);
g.fillRect(0, 0, 3, 592);
```

```
g.fillRect(0, 0, 692, 3);
g.fillRect(691, 0, 3, 592);

// Paddle
g.setColor(Color.green);
g.fillRect(playerX, 550, 100, 8);

// Ball
g.setColor(Color.yellow);
g.fillOval(ballX, ballY, 20, 20);

// Bricks
for (int i = 0; i < bricks.length; i++) {
for (int j = 0; j < bricks[0].length; j++) {
if (bricks[i][j] > 0) {
g.setColor(Color.red);
g.fillRect(j * 100 + 80, i * 50 + 50, 80, 30);
g.setColor(Color.black);
g.drawRect(j * 100 + 80, i * 50 + 50, 80, 30);
}
}
```

```
}

// Score
g.setColor(Color.white);
g.setFont(new Font("serif", Font.BOLD, 25));
g.drawString("Score: " + score, 590, 30);

// Game Over
if (totalBricks <= 0) {
    play = false;
    ballXDir = 0;
    ballYDir = 0;
    g.setColor(Color.green);
    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);
}

if (ballY > 570) {
    play = false;
```

```
ballXDir = 0;
ballYDir = 0;
g.setColor(Color.red);
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) {
// Ball-Paddle collision
if (new Rectangle(ballX, ballY, 20, 20).intersects(new
Rectangle(playerX, 550, 100, 8))) {
ballYDir = -ballYDir;
}
```

```
// Ball-Brick collision
A: for (int i = 0; i < bricks.length; i++) {
for (int j = 0; j < bricks[0].length; j++) {
if (bricks[i][j] > 0) {
int brickX = j * 100 + 80;
int brickY = i * 50 + 50;
int brickWidth = 80;
int brickHeight = 30;

Rectangle rect = new Rectangle(brickX, brickY,
brickWidth, brickHeight);
Rectangle ballRect = new Rectangle(ballX, ballY, 20,
20);

if (ballRect.intersects(rect)) {
bricks[i][j] = 0;
totalBricks--;
score += 5;

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

```
ballXDir = -ballXDir;
} else {
ballYDir = -ballYDir;
}
break A;
}
}
}
}

ballX += ballXDir;
ballY += ballYDir;
if (ballX < 0) {
ballXDir = -ballXDir;
}
if (ballY < 0) {
ballYDir = -ballYDir;
}
if (ballX > 670) {
ballXDir = -ballXDir;
}
```

```
}
repaint();
}

@Override
public void keyTyped(KeyEvent e) {}

@Override
public void keyReleased(KeyEvent e) {}

@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;
ballX = 200;
ballY = 300;
ballXDir = -1;
ballYDir = -2;
playerX = 310;
score = 0;
totalBricks = 21;
bricks = new int[3][7];
for (int i = 0; i < bricks.length; i++) {
for (int j = 0; j < bricks[0].length; j++) {
bricks[i][j] = 1;
}
}
```

```
repaint();
}
}
}

public void moveRight() {
play = true;
playerX += 20;
}

public void moveLeft() {
play = true;
playerX -= 20;
}

public static void main(String[] args) {
JFrame frame = new JFrame("Brick Breaker");
BrickBreaker gamePlay = new BrickBreaker();
frame.setBounds(10, 10, 700, 600);
frame.setResizable(false);
frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.add(gamePlay);
}
}
```

Minimum Hardware/Software Requirements

Hardware: Any computer with Java installed will work. A minimum of 1 GB of RAM is required with a simple processor.

- **Software: Java compiler to be installed in the system; text editors or IDEs like VS Code and IntelliJ Idea to write and run the script.**

References

JAVA Documentation: https://docs.oracle.com/en/java/

Brick Breaker in JAVA:

https://www.javatpoint.com/building-a-brick-breaker-game-in-java

Input and Output Handling in JAVA:

https://www.javatpoint.com/java-io

CONCLUSION

In conclusion, the Brick Breaker game made in Java is a fun and engaging game that demonstrates the power of Java programming in game development. With its simple yet addictive gameplay, it is an excellent starting point for beginners looking to learn game development with Java. By enhancing the game with additional features, such as multiple levels, power-ups, and AI opponents, we can create an even more exciting and challenging game that will keep players engaged for hours.

BIBLIOGRAPHY

Books:

- **Java: A Beginner's Guide by Herbert Schildt**
- **Java: The Complete Reference by Herbert Schildt**
- **Game Programming with Java by John Horton**
- **Java Game Development with LibGDX by Lee Stemkoski**

Research Papers:

- **A Study on Game Development using Java by S. S. Rao**
- **Java-based Game Development: A Review by A. K. Singh**

Online Resources

- **Oracle Java Tutorials. (n.d.). Retrieved from https://docs.oracle.com/javase/tutorial/**
- **Java Game Development Tutorial by Tutorials Point. (n.d.). Retrieved from https://www.tutorialspoint.com/java/java_game_development.htm**
- **Brick Breaker Game Tutorial by Java Tutorials Point. (n.d.). Retrieved from https://www.tutorialspoint.com/java/java_brick_breaker_game.htm**
- **Java Game Development with JavaFX by Oracle. (n.d.). Retrieved from https://docs.oracle.com/javafx/2/game-dev-tutorial/index.html**