

SNAKE GAME IN JAVA

A PROJECT REPORT

Submitted by

NAMETHA K 2303811724322074

in partial fulfillment of requirements for the award of the course

CGB1201 – JAVA PROGRAMMING

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by
AICTE, New Delhi)

SAMAYAPURAM – 621 112

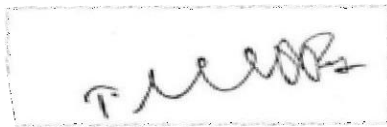
DECEMBER, 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “ **SNAKE GAME IN JAVA**” is the bonafide work of **NAMETHA K 2303811724322074** who carried out the project work during the academic year 2024 - 2025 under my supervision.



Signature

Dr. T. AVUDAIAPPAN M.E.,Ph.D.,

HEAD OF THE DEPARTMENT,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.



Signature

Mrs.S.Geetha M.E.,


SUPERVISOR,

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.

Submitted for the viva-voce examination held on 3.12.24



INTERNAL EXAMINER

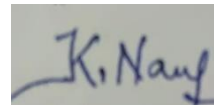


EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**SNAKE GAME IN JAVA**” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.

Signature



NAMETHA K

Place: Samayapuram

Date: 3/12/2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **“K. Ramakrishnan College of Technology (Autonomous)”**, for providing us with the opportunity to do this project.

I extend our sincere acknowledgement and appreciation to the esteemed and honourable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S.KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E.**, Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards.

MISSION OF THE INSTITUTION

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

VISION AND MISSION OF THE DEPARTMENT

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO 1: Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

PEO 2: Provide industry-specific solutions for the society with effective communication and ethics

PEO 3: Hone their professional skills through research and lifelong learning initiatives.

PROGRAM OUTCOMES

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- 5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- 6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- 7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- 8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

- 9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- 10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- 11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- 12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

ABSTRACT

This Java program implements a simple Snake Game using the Abstract Window Toolkit (AWT) for graphical user interface and game rendering. It utilizes basic game mechanics such as collision detection, movement logic, and scoring. The game operates on a grid-based system and runs in a separate thread to enable smooth and consistent gameplay. The player controls the snake using the arrow keys to consume food, which increases the snake's length and score, while avoiding collisions with walls or the snake's body.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
	ABSTRACT	viii
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.2 OBJECTIVE	1
2	PROJECT METHODOLOGY	2
	2.1 PROPOSED WORK	2
	2.2 BLOCK DIAGRAM	3
3	JAVA PROGRAMMING CONCEPTS	4
	3.1 Abstract Window Toolkit (AWT)	4
	3.2 Event Handling	4
4	MODULE DESCRIPTION	6
	4.1 GAME SETUP	6
	4.2 SNAKE MOVEMENT	6
	4.3 FOOD PLACEMENT	6
	4.4 COLLISION DETECTION	6
	4.5 GRAPHICS	6
5	CONCLUSION	7
	REFERENCES	8
	APPENDICES	9
	Appendix A – SOURCE CODE	9
	Appendix B – SCREEN SHOTS	14

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In this implementation, the player controls a snake that navigates a grid-based board, consuming randomly placed food to grow longer and earn points. The challenge lies in avoiding collisions with the snake's own body or the walls of the game board, which would end the game. The program uses keyboard input for real-time interaction, rendering the snake and food dynamically using Java's Graphics class. By integrating game mechanics such as continuous movement, dynamic rendering, and game state management, this project serves as a practical example for beginners and enthusiasts to explore game development using Java AWT. The game's modular design allows for easy understanding, modification, and potential enhancements.

1.2 OBJECTIVE

The objective of this program is to create an interactive and engaging Snake Game using Java and the Abstract Window Toolkit (AWT), replicating the core mechanics of the classic arcade game. The game focuses on smooth snake movement, controlled by real-time keyboard inputs, as players navigate the snake to consume randomly placed food, increasing its length and score. The program tracks and displays the player's score, encouraging competitiveness. Beyond entertainment, this program demonstrates fundamental programming concepts such as event-driven input handling, threading for game loops, and dynamic rendering using AWT's Graphics class. It serves as a practical example for learners and developers exploring 2D game development in Java.

CHAPTER 2

PROJECT METHODOLOGY

2.1 PROPOSED WORK

The proposed work focuses on designing and implementing a 2D Snake Game using Java's Abstract Window Toolkit (AWT) to deliver an engaging and interactive experience. The game will feature a grid-based board where players control a snake that moves to consume randomly placed food, growing in length and earning points. Snake movement will be managed using an array-based system, with real-time direction changes enabled through keyboard inputs handled by `KeyListener`. The game will include robust collision detection to handle scenarios where the snake collides with walls or its own body, resulting in a game-over state. A separate thread will manage the game loop to ensure smooth updates for movement, rendering, and collision checks. The interface will display the current score and provide a "Game Over" screen with the option to restart the game seamlessly, enhancing replayability. This project aims to combine intuitive gameplay with essential programming concepts like event-driven input handling, threading, and dynamic rendering to create a functional and visually appealing application.

2.2 BLOCK DIAGRAM

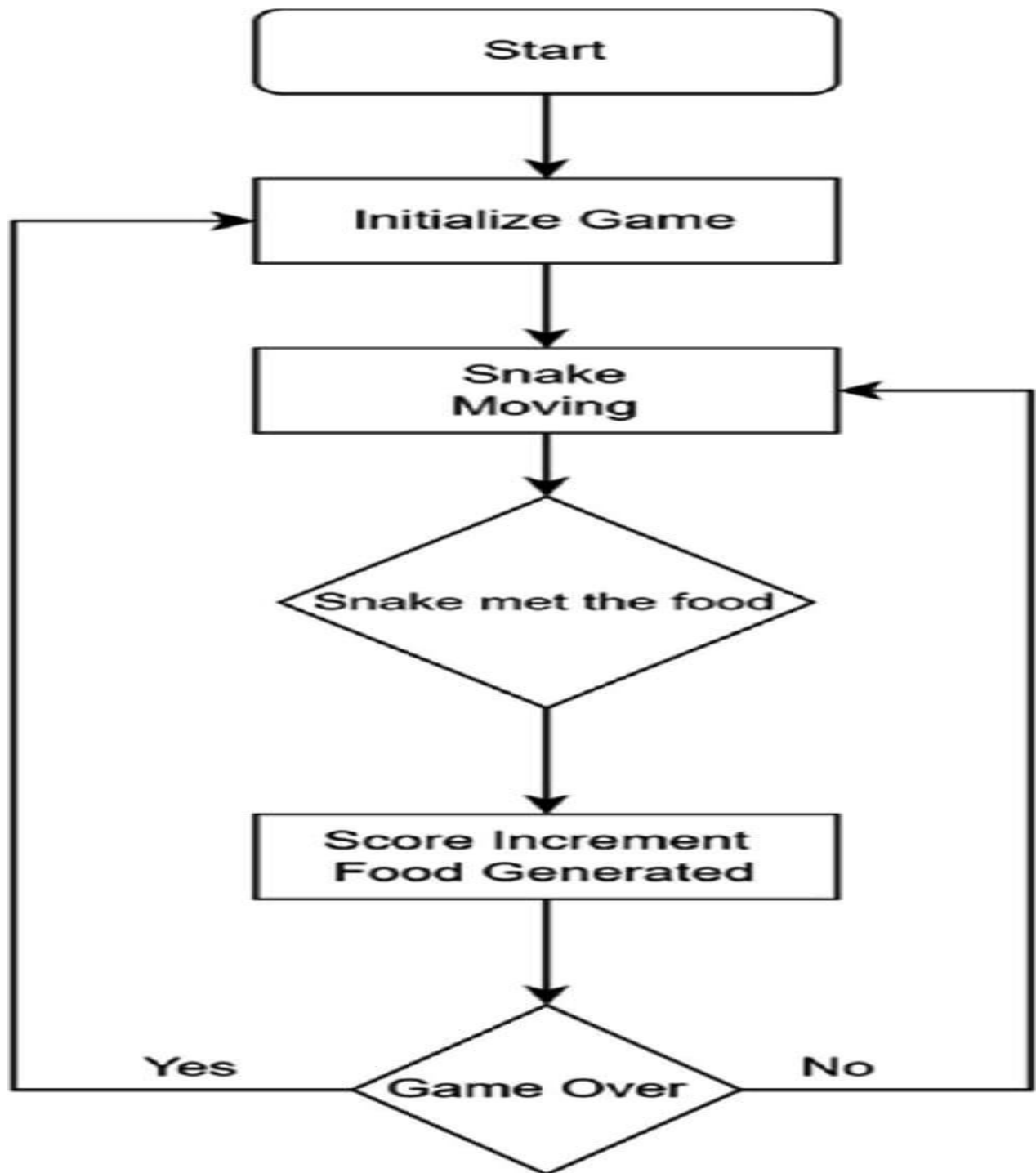


Fig 1.1 (Snake Game Flow Diagram)

CHAPTER 3

JAVA PROGRAMMING CONCEPTS

3.1 Abstract Window Toolkit (AWT):

The Abstract Window Toolkit (AWT) is a platform-independent toolkit used for building graphical user interfaces in Java. In this game, the Frame class is used to create the main game window, while the Graphics class handles rendering of the game elements such as the snake, food, and score. The Color class is used to customize the appearance of these elements. The paint(Graphics g) method is overridden to draw the snake's body using rectangles (fillRect) and the food using ovals (fillOval). AWT provides the foundation for interactive and visually engaging applications.

3.2 Event Handling:

Event handling in Java is implemented using Event Listeners to capture user actions. The game utilizes the KeyListener interface to detect keyboard inputs, allowing players to change the snake's direction. Methods like keyPressed(KeyEvent e) are overridden to process arrow key presses and update the movement logic in real time. Additionally, a WindowAdapter is used to handle the windowClosing event, ensuring that the game exits gracefully when the user closes the window.

Multithreading:

Multithreading allows concurrent execution of tasks, ensuring smooth performance. The game uses a separate Thread to manage the game loop, which handles continuous updates for movement, collision checks, and rendering. The run() method defines the logic executed by this thread. To control the game's speed, Thread.sleep(100) is used to introduce a delay between frames. This creates a consistent refresh rate, ensuring the game is neither too fast nor too slow. Multithreading ensures the game updates without freezing the UI.

Arrays for Snake Representation

Arrays are a fundamental data structure used to store the coordinates of the snake's body. Two arrays, x and y, represent the x and y positions of each segment of the snake. As the snake grows, new segments are added by increasing the snakeLength variable. During each update, the coordinates of the snake's body are shifted to reflect its movement, with the head being updated first. This use of arrays makes it efficient to manage the snake's dynamic growth and movement.

The Random class is used to generate random positions for the food on the game board. The nextInt() method ensures the food is placed within the bounds of the board and aligned with the grid size. For example:

```
foodX = random.nextInt(BOARD_WIDTH / UNIT_SIZE) * UNIT_SIZE;
```

```
foodY = random.nextInt(BOARD_HEIGHT / UNIT_SIZE) * UNIT_SIZE;
```

Randomization ensures that the food appears in new locations after each consumption, adding variety and challenge to the gameplay. This concept demonstrates how randomness can enhance the user experience in games. These concepts work together to create a fully functional and interactive game, leveraging Java's core features to handle GUI, input, logic, and rendering seamlessly.

CHAPTER 4

MODULE DESCRIPTION

4.1 Game Setup

Window Creation: Frame is used to set up the game window with a fixed size and visibility.

Key Listener: Listens for keyboard input to control the snake.

4.2 Snake Movement

Positioning: Arrays x[] and y[] store the snake's body coordinates.

Direction Control: The direction variable (R, L, U, D) determines movement.

Move Logic: The snake's body shifts positions, with the head leading

4.3 Food Placement

Random Generation: Food is placed at random coordinates within game board

Collision Check: If the snake eats the food, it grows, and the score increases

4.4 Collision Detection

Wall Collision: Ends the game if the snake hits the board boundaries.

Body Collision: Ends the game if the snake collides with itself

4.5 Graphics

Rendering: The paint() method draws the snake, food, and score.

Game Over: Displays a "Game Over" message when the game ends.

CHAPTER 5

CONCLUSION

The Snake Game implemented using Java's Abstract Window Toolkit (AWT) successfully recreates the classic gameplay while demonstrating essential programming concepts such as event-driven input handling, threading, and graphical rendering. The project provides an engaging user experience by incorporating smooth snake movement, responsive controls, dynamic food placement, and effective collision detection. By displaying the player's score and offering a seamless restart option, the game encourages replayability and competitiveness. Additionally, the program serves as a practical example of how fundamental Java features can be used to develop interactive 2D games. This implementation not only fulfills its goal of delivering a functional and enjoyable game but also provides a solid foundation for learners and developers to explore advanced game development techniques.

REFERENCES:

1. Java Platform Documentation

Oracle. "Abstract Window Toolkit (AWT)."

<https://docs.oracle.com/javase/8/docs/api/java/awt/package-summary.html>

Comprehensive documentation on Java AWT, covering its classes and methods for building graphical user interfaces.

2. Java Tutorials

Oracle. "The Java™ Tutorials: AWT and Swing."

<https://docs.oracle.com/javase/tutorial/uiswing/>

A beginner-friendly guide to using AWT and Swing for creating GUI applications.

3. Game Development Concepts

Buckland, Mat. Programming Game AI by Example. Wordware Publishing, 2005.

A practical resource on game development, including grid-based movement and collision detection principles.

APPENDICES

APPENDIX A – SOURCE CODE

```
import java.awt.*;
import java.awt.event.*;
import java.util.Random;

public class SnakeGameAWT extends Frame implements Runnable, KeyListener {
    private static final int BOARD_WIDTH = 800;
    private static final int BOARD_HEIGHT = 600;
    private static final int UNIT_SIZE = 25;

    private final int[] x = new int[BOARD_WIDTH * BOARD_HEIGHT /
UNIT_SIZE];
    private final int[] y = new int[BOARD_WIDTH * BOARD_HEIGHT /
UNIT_SIZE];
    private int snakeLength;
    private int foodX, foodY;
    private int score;

    private char direction;
    private boolean running;
    private boolean gameOver;

    private Thread gameThread;

    public SnakeGameAWT() {
        // Set up the frame
        this.setTitle("Snake Game - AWT");
        this.setSize(BOARD_WIDTH, BOARD_HEIGHT);
        this.setResizable(false);
        this.setVisible(true);
        this.addKeyListener(this);

        // Start the game
        startGame();

        // Close the application on window close
        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

```

    }
    });
}

public void startGame() {
    // Initialize snake's position and properties
    snakeLength = 3;
    direction = 'R'; // Initial direction
    score = 0;
    running = true;
    gameOver = false;

    for (int i = 0; i < snakeLength; i++) {
        x[i] = 50 - i * UNIT_SIZE;
        y[i] = 50;
    }

    placeFood();

    // Start the game thread
    if (gameThread == null || !gameThread.isAlive()) {
        gameThread = new Thread(this);
        gameThread.start();
    }
}

public void placeFood() {
    Random random = new Random();
    foodX = random.nextInt(BOARD_WIDTH / UNIT_SIZE) * UNIT_SIZE;
    foodY = random.nextInt(BOARD_HEIGHT / UNIT_SIZE) * UNIT_SIZE;
}

public void checkFoodCollision() {
    if (x[0] == foodX && y[0] == foodY) {
        snakeLength++;
        score += 10;
        placeFood();
    }
}

public void checkWallCollision() {
    if (x[0] < 0 || x[0] >= BOARD_WIDTH || y[0] < 0 || y[0] >= BOARD_HEIGHT)
{

```

```

        running = false;
        gameOver = true;
    }
}

public void checkBodyCollision() {
    for (int i = 1; i < snakeLength; i++) {
        if (x[0] == x[i] && y[0] == y[i]) {
            running = false;
            gameOver = true;
        }
    }
}

public void move() {
    for (int i = snakeLength; i > 0; i--) {
        x[i] = x[i - 1];
        y[i] = y[i - 1];
    }
    switch (direction) {
        case 'R' -> x[0] += UNIT_SIZE;
        case 'L' -> x[0] -= UNIT_SIZE;
        case 'U' -> y[0] -= UNIT_SIZE;
        case 'D' -> y[0] += UNIT_SIZE;
    }
}

@Override
public void run() {
    while (true) {
        if (running) {
            move();
            checkFoodCollision();
            checkWallCollision();
            checkBodyCollision();
            repaint();
        }

        try {
            Thread.sleep(100); // Game speed
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
}

@Override
public void paint(Graphics g) {
    if (running) {
        // Draw food
        g.setColor(Color.RED);
        g.fillOval(foodX, foodY, UNIT_SIZE, UNIT_SIZE);

        // Draw snake
        for (int i = 0; i < snakeLength; i++) {
            if (i == 0) {
                g.setColor(Color.GREEN); // Snake's head
            } else {
                g.setColor(Color.BLACK); // Snake's body
            }
            g.fillRect(x[i], y[i], UNIT_SIZE, UNIT_SIZE);
        }

        // Draw score
        g.setColor(Color.BLUE);
        g.setFont(new Font("Arial", Font.BOLD, 20));
        g.drawString("Score: " + score, 10, 50);
    } else if (gameOver) {
        // Game Over screen
        g.setColor(Color.RED);
        g.setFont(new Font("Arial", Font.BOLD, 50));
        g.drawString("Game Over", BOARD_WIDTH / 3, BOARD_HEIGHT / 3);

        g.setFont(new Font("Arial", Font.BOLD, 20));
        g.drawString("Press Enter to Restart", BOARD_WIDTH / 3,
BOARD_HEIGHT / 2);
    }
}

@Override
public void keyPressed(KeyEvent e) {
    if (running) {
        switch (e.getKeyCode()) {
            case KeyEvent.VK_LEFT -> {
                if (direction != 'R') direction = 'L';
            }
        }
    }
}

```

```

        case KeyEvent.VK_RIGHT -> {
            if (direction != 'L') direction = 'R';
        }
        case KeyEvent.VK_UP -> {
            if (direction != 'D') direction = 'U';
        }
        case KeyEvent.VK_DOWN -> {
            if (direction != 'U') direction = 'D';
        }
    }
} else if (gameOver && e.getKeyCode() == KeyEvent.VK_ENTER) {
    // Restart the game
    startGame();
    repaint();
}
}

@Override
public void keyReleased(KeyEvent e) {}

@Override
public void keyTyped(KeyEvent e) {}

public static void main(String[] args) {
    new SnakeGameAWT();
}
}

```

APPENDIX B - SCREENSHOTS

