

SNAKE GAME



A PROJECT REPORT

Submitted by

SHERWIN B (2303811710421146)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

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

SAMAYAPURAM – 621 112

NOVEMBER- 2024

SAMAYAPURAM – 621 112

Certified that this project report on “**SNAKE GAME**” is the bonafide work of **SHERWIN B (2303811710421146)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Mr. MALA MANNAN A, M.E.,
J. Madam
ASSISTANT PROFESSOR

SIGNATURE

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram-621112.

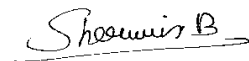
CGB1201-JAVA PROGRAMMING
Mr.R. KARUNIK, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8138-SCE, TRICHY.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**SNAKE GAME** ” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

Signature



SHERWIN B

Place: Samayapuram

Date: 6.12.24

ACKNOWLEDGEMENT

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

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for 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 to our project and offering adequate duration in completing our project.

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

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E.,Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **MR. A. MALARMANNAN, M.E.**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

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

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

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.

ABSTRACT

The Snake Game is a classic arcade game that has been implemented in Java using the Swing library for graphical user interface (GUI). The game involves controlling a snake that moves within a grid, consuming food to grow in size while avoiding collisions with walls and itself. The project emphasizes fundamental programming concepts such as event-driven programming, object-oriented design, and real-time game logic.

The implementation uses a structured approach to design a responsive and interactive game. Key components include a JPanel for rendering the game screen, a Timer for controlling the game loop, and a KeyListener for handling user input. The snake's movement, food spawning, and collision detection are efficiently managed using arrays and logical conditions. The game dynamically updates the score as the snake consumes food, providing real-time feedback to the player.

This project serves as an excellent introduction to game development in Java, offering a foundation for exploring more advanced features such as enhanced graphics, multiple levels, and high-score tracking. The modular design ensures code reusability and scalability, making it a valuable resource for beginners and enthusiasts interested in building interactive applications.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>The Snake Game is a classic arcade game that has been implemented in Java using the Swing library for graphical user interface (GUI). The game involves controlling a snake that moves within a grid, consuming food to grow in size while avoiding collisions with walls and itself. The project emphasizes fundamental programming concepts such as event-driven programming, object-oriented design, and real-time game logic. The implementation uses a structured approach to design a responsive and interactive game. Key components include a JPanel for rendering the game screen, a Timer for controlling the game loop, and a KeyListener for handling user input. The snake's movement, food spawning, and collision detection are efficiently managed using arrays and logical conditions. The game dynamically updates the score as the snake consumes food, providing real-time feedback to the player.</p>	<p>PO1 -3 PO2 -3 PO3 -3 PO5 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3</p>	<p>PSO1 -3 PSO2 -3 PSO3 -3</p>

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	
1	INTRODUCTION	
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
2	PROJECT METHODOLOGY	
	2.1 Proposed Work	3
	2.2 Block Diagram	4
3	MODULE DESCRIPTION	
	3.1 Game Initialization Module	5
	3.2 Game Loop Management Module	5
	3.3 Snake Movement and Control Module	5
	3.4 Collision Detection Module	5
	3.5 Food Generation and Scoring Module	5
4	CONCLUSION & FUTURE SCOPE	
	4.1 Conclusion	6
	4.2 Future Scope	7
	REFERENCES	
	APPENDIX A (SOURCE CODE)	8
	APPENDIX B (SCREENSHOTS)	12

CHAPTER 1

INTRODUCTION

1.1 Objective

The primary objective of the Snake game project is to develop an interactive and engaging application in Java using the principles of object-oriented programming (OOP). The game focuses on implementing a simple yet effective structure to handle gameplay mechanics such as snake movement, collision detection, food generation, and scoring. It provides a foundation for understanding graphical programming, event handling, and game logic design in Java.

1.2 Overview

The Snake game is a classic arcade-style game where a player controls a snake navigating a grid. The goal is to eat food items that randomly appear on the grid, causing the snake to grow longer. The player must avoid colliding with the walls or the snake's own body to keep the game running.

This project is developed using Java Swing for graphical rendering and key handling, and it employs OOP principles to ensure modular, maintainable, and scalable code. The gameplay includes:

1. A continuously updating game loop.
2. User input for controlling the snake's direction.
3. Dynamic food generation and collision detection.
4. Scoring mechanisms.
5. Endgame conditions when a collision occurs.

1.3 Java Programming Concepts

Basic OOP Concepts

1. Encapsulation:
 - The game encapsulates data and behavior within classes such as GameBoard, which manages game state, and SnakeGame, which initializes the application window.
 - The snake's state (position and size) is encapsulated in a LinkedList data structure.
2. Inheritance:
 - The GameBoard class inherits from JPanel to handle rendering and event handling.
3. Polymorphism:
 - Event handling in the GameBoard class uses polymorphism by overriding methods such as actionPerformed for ActionListener and keyPressed for KeyAdapter.
4. Abstraction:
 - Complex operations like rendering the board, checking collisions, and updating the game state are abstracted into methods to simplify readability and maintainability.

Project-Specific Java Concepts

1. Swing Framework:
 - Used to create the game window and handle graphical rendering through JPanel and JFrame.
 - Components like Timer are used for a consistent game loop.
2. Event Handling:

- The KeyListener is implemented to capture keyboard inputs for controlling the snake's direction.
3. Collections Framework:
 - The snake is represented using a LinkedList of Point objects to dynamically adjust its size and manage positions.
 4. Randomization:
 - The Random class is used for generating food at random positions on the grid while avoiding overlapping with the snake.
 5. Game Loop Management:
 - The game loop is controlled using a Timer, which ensures periodic updates to the game state and UI.
 6. 2D Graphics:
 - The Graphics class is used to draw the game board, the snake, and food items on the screen.

These concepts collectively demonstrate how OOP principles and Java-specific features are effectively used to implement a functional and interactive Snake game.

CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for developing the Snake game involves designing and implementing an interactive application using Java. The primary focus is on leveraging object-oriented programming principles and Java's graphical capabilities to create a structured, functional, and visually appealing game.

2.2 Block Diagram

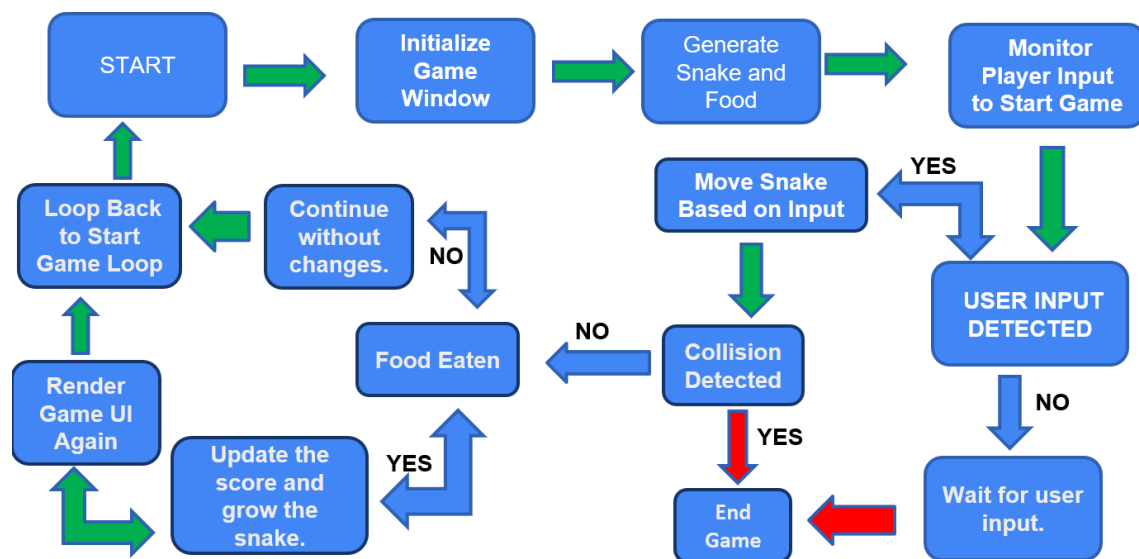


Fig – 2.1 Block Diagram

CHAPTER 3

MODULE DESCRIPTION

3.1 Module 1: Game Initialization

- This module is responsible for setting up the game environment. It includes the creation of the game window, setting up the grid dimensions, initializing the snake's starting position, and spawning the first food item.

3.2 Module 2: Game Loop Management

- This module handles the periodic updates to the game state. It ensures smooth gameplay by managing the flow of time using a timer and coordinating the movement of the snake, checking collisions, and refreshing the game interface.

3.3 Module 3: Snake Movement and Controls

- This module manages the snake's movement across the grid. It listens to user input via keyboard events and updates the snake's direction accordingly. It also calculates the new position of the snake and ensures that the body follows the head correctly.

3.4 Module 4: Collision Detection

- This module checks for various game-ending conditions, such as the snake colliding with itself or the boundaries of the grid. It ensures that these events are accurately detected and triggers the endgame sequence when necessary.

3.5 Module 5: Food Generation and Scoring

- This module is responsible for spawning food at random positions on the grid, ensuring it does not overlap with the snake. It also tracks the score and increases the snake's length upon consuming food. The score is updated and displayed dynamically on the interface.

These modules work together to deliver a cohesive, interactive, and enjoyable Snake game experience. Each module encapsulates specific responsibilities, ensuring modularity and ease of maintenance.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The Snake game project successfully combines programming concepts, game logic, and user interface design into an engaging and functional application. The following points summarize the key outcomes and insights gained from the project:

1. Demonstration of OOP Principles

- The project highlights the use of object-oriented programming concepts such as **encapsulation**, **inheritance**, **polymorphism**, and **abstraction** to design a structured and modular system.

2. Implementation of Game Logic

- The game's core logic, including smooth snake movement, collision detection, and food generation, was implemented effectively, ensuring an engaging user experience.

3. Graphical Programming in Java

- The use of Java Swing and the Graphics class allowed for the creation of a visually appealing and interactive graphical user interface, introducing concepts of 2D rendering and real-time updates.

4. Modular Design

- The separation of functionality into distinct modules, such as game initialization, movement, collision detection, and scoring, ensures scalability, maintainability, and ease of debugging.

5. Scalability and Extendability

- The project serves as a robust foundation for future enhancements. Features like levels of difficulty, obstacles, multiplayer modes, and advanced animations can be seamlessly integrated into the existing architecture.

The Snake game project not only delivers a complete and functional application but also

serves as a stepping stone for aspiring developers to delve deeper into game development and Java-based application design. With its modular structure and extendability, the project opens doors to exploring more advanced concepts in software and game engineering

4.2 FUTURE SCOPE

The future scope for a Snake Game in Java includes numerous enhancements to improve gameplay, user engagement, and its relevance in modern applications. Gameplay features can be expanded with multiplayer modes, dynamic obstacles, power-ups, and customizable snakes, while levels and progression systems can enhance the challenge. Upgrading the graphics with 3D designs, smooth animations, and thematic environments using libraries like JavaFX can modernize the game. It can also serve educational purposes by demonstrating algorithms like pathfinding or teaching programming concepts such as object-oriented programming and data structures. Integrating emerging technologies like AI for adaptive gameplay, AR for real-world interaction, VR for immersive experiences, and IoT for unconventional controls adds an innovative edge. Social and competitive features, including leaderboards, achievements, and social sharing, can make the game more engaging. Gamification for specific audiences, such as educational games, fitness-based controls, or accessibility features, can cater to diverse user groups. Overall, the Snake Game in Java holds significant potential for development as a modern, educational, and entertaining application.

CHAPTER 5 APPENDIX A

(SOURCE CODE)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.Random;

public class SnakeGame extends JFrame {
    public SnakeGame() {
        this.add(new GamePanel());
        this.setTitle("Snake Game");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(false);
        this.pack();
        this.setLocationRelativeTo(null);
        this.setVisible(true);
    }

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

class GamePanel extends JPanel implements ActionListener {
    private final int TILE_SIZE = 25; // Size of each tile
    private final int GAME_WIDTH = 500;
    private final int GAME_HEIGHT = 500;
    private final int TOTAL_TILES = (GAME_WIDTH * GAME_HEIGHT) /
        (TILE_SIZE * TILE_SIZE);
    private final int[] x = new int[TOTAL_TILES];
    private final int[] y = new int[TOTAL_TILES];
    private int bodyParts = 3; // Initial snake size
    private int foodX, foodY; // Food position
    private int score = 0;
    private char direction = 'R'; // Initial direction: R = right
    private boolean running = false;
    private Timer timer;
    private Random random;

    public GamePanel() {
```

```

        this.setPreferredSize(new Dimension(GAME_WIDTH, GAME_HEIGHT));
        this.setBackground(Color.BLACK);
        this.setFocusable(true);
        this.addKeyListener(new MyKeyAdapter());
        random = new Random();
        startGame();
    }

    private void startGame() {
        spawnFood();
        running = true;
        timer = new Timer(100, this); // Speed of the game
        timer.start();
    }

    private void spawnFood() {
        foodX = random.nextInt(GAME_WIDTH / TILE_SIZE) * TILE_SIZE;
        foodY = random.nextInt(GAME_HEIGHT / TILE_SIZE) * TILE_SIZE;
    }

    private void move() {
        for (int i = bodyParts; i > 0; i--) {
            x[i] = x[i - 1];
            y[i] = y[i - 1];
        }

        switch (direction) {
            case 'U' -> y[0] -= TILE_SIZE;
            case 'D' -> y[0] += TILE_SIZE;
            case 'L' -> x[0] -= TILE_SIZE;
            case 'R' -> x[0] += TILE_SIZE;
        }
    }

    private void checkFood() {
        if ((x[0] == foodX) && (y[0] == foodY)) {
            bodyParts++;
            score++;
            spawnFood();
        }
    }

    private 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 touches walls
if (x[0] < 0 || x[0] >= GAME_WIDTH || y[0] < 0 || y[0] >= GAME_HEIGHT) {
    running = false;
}

if (!running) {
    timer.stop();
}
}

private void gameOver(Graphics g) {
    g.setColor(Color.RED);
    g.setFont(new Font("Arial", Font.BOLD, 40));
    FontMetrics metrics = getFontMetrics(g.getFont());
    g.drawString("Game Over", (GAME_WIDTH - metrics.stringWidth("Game
        Over")) / 2, GAME_HEIGHT / 2);

    g.setColor(Color.WHITE);
    g.setFont(new Font("Arial", Font.PLAIN, 20));
    g.drawString("Score: " + score, (GAME_WIDTH - metrics.stringWidth("Score: "
        + score)) / 2, GAME_HEIGHT / 2 + 50);
}

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    if (running) {
        g.setColor(Color.RED);
        g.fillOval(foodX, foodY, TILE_SIZE, TILE_SIZE);

        for (int i = 0; i < bodyParts; i++) {
            if (i == 0) {
                g.setColor(Color.GREEN);
            } else {
                g.setColor(new Color(45, 180, 0));
            }
            g.fillRect(x[i], y[i], TILE_SIZE, TILE_SIZE);
        }

        g.setColor(Color.WHITE);
        g.setFont(new Font("Arial", Font.PLAIN, 20));
        g.drawString("Score: " + score, 10, 20);
    }
}

```

```

        } else {
            gameOver(g);
        }
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (running) {
            move();
            checkFood();
            checkCollisions();
        }
        repaint();
    }

    private class MyKeyAdapter extends KeyAdapter {
        @Override
        public void keyPressed(KeyEvent e) {
            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';
                }
            }
        }
    }
}

```

APPENDIX B (SCREENSHOTS)



REFERENCES

References

1. Java Documentation
 - Official Java Platform Documentation by Oracle.
<https://docs.oracle.com/javase/>
2. Java Swing Tutorials
 - Comprehensive tutorials for GUI development using Swing.
<https://docs.oracle.com/javase/tutorial/uiswing/>
3. OOP Principles and Design Patterns
 - A resource on object-oriented programming principles and how they are applied in game development.
Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software.
4. Event Handling in Java
 - Java Tutorials on implementing keyboard and action listeners.
https://www.tutorialspoint.com/java/java_event_handling.htm
5. Graphics Programming in Java
 - A guide to rendering 2D graphics in Java using the Graphics class.
<https://www.geeksforgeeks.org/graphics-in-java/>
6. Randomization in Java
 - Tutorials on generating random numbers and their application in game design.
<https://www.baeldung.com/java-generating-random-numbers>
7. Game Development Projects
 - Snake Game examples and inspirations from open-source repositories on GitHub.
<https://github.com/>
8. Books on Java Programming
 - *Schildt, H. (2018). Java: The Complete Reference (11th Edition).* McGraw-Hill Education.