A Project Report

on

Snake game using Java

Submitted in partial fulfillment of requirements for the award of the course

of

**CGB1201 – JAVA PROGRAMMING**

Under the guidance of

### Mrs. P. GEETHA, M.E.,

Submitted By



**HARISH KANNA R (8115U23AM020)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

(ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING**

(Autonomous)

TRICHY-621 112

DECEMBER 2024

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING**

**(Autonomous Institution affiliated to Anna University, Chennai)**

**TRICHY-621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on **“ SNAKE GAME USING JAVA”** is the bonafide work of **HARISH KANNA R (8115U23AM020)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

|  |  |
| --- | --- |
| Signature | Signature |
| **Mr. P.GEETHA,** | **Dr. B.KIRAN BALA** |
| **ASSISTANT PROFESSOR,** | **HEAD OF THE DEPARTMENT,** |
| Department of Artificial Intelligence and Data Science, | Department of Artificial Intelligence and Data Science, |
| K. Ramakrishnan College of Engineering, | K. Ramakrishnan College of Engineering, |
| Samayapuram, Trichy-621 112.  . | Samayapuram, Trichy-621 112. |

Submitted for the End Semester Examination held on …………….

**INTERNAL EXAMINER EXTERNAL EXAMINER**

## DECLARATION

## I jointly declare that the project report on “SNAKE GAME USING JAVA” 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 fulfillment of the requirement of the award of the course CGB1201- JAVA PROGRAMMING

## SIGNATURE

## HARISH KANNA R

## ACKNOWLEDGEMENT

## It is with great pride that I express our gratitude and indebtedness to our institution,“K.RAMAKRISHNAN COLLEGE OF ENGINEERING

## (Autonomous)”,for providing us with the opportunity to do this project. I extend our sincere acknowledgment and appreciation to the esteemed and honorable 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. D. SRINIVASAN, M.E., Ph.D., FIE., MIIW.,MISTE., MISAE., C. Engg., Principal, who gave the opportunity to frame the project to full satisfaction.

## I would like to thank Dr. B. KIRAN BALA, B.Tech., M.E., M.B.A., Ph.D., M.I.S.T.E., U.A.C.E.E., IAENG, Head of the Department of Artificial Intelligence and Machine Learning, for providing his encouragement in pursuing this project.

## I wish to convey our profound and heartfelt gratitude to our esteemed project guide Mr.P.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

## INSTITUTE VISION AND MISSION

**VISION OF THE INSTITUTE:**

To achieve a prominent position among the top technical institutions.

**MISSION OF THE INSTIITUTE:**

**M1:** To best owstandard technical education parexcellence through state of the art infrastructure, competent faculty and high ethical standards.

**M2:** To nurture research and entrepreneurial skills among students in cutting edge technologies.

**M3:** To provide education for developing high-quality professionals to transform the society.

## DEPARTMENT VISION AND MISSION

**DEPARTMENT OF CSE(ARTIFICIAL INTELLIGENCE AND MACHINELEARNING)**

**Vision of the Department**

To become a renowned hub for Artificial Intelligence and Machine Learning

Technologies to produce highly talented globally recognizable technocrats to meet

Industrial needs and societal expectations.

**Mission of the Department**

**M1**: To impart advanced education in Artificial Intelligence and Machine Learning,

Built upon a foundation in Computer Science and Engineering.

**M2**: To foster Experiential learning equips students with engineering skills to

Tackle real-world problems.

**M3**: To promote collaborative innovation in Artificial Intelligence, machine

Learning, and related research and development with industries.

**M4**: To provide an enjoyable environment for pursuing excellence while upholding

Strong personal and professional values and ethics.

**Programme Educational Objectives (PEOs):**

Graduates will be able to:

**PEO1**: Excel in technical abilities to build intelligent systems in the fields of

Artificial Intelligence and Machine Learning in order to find new opportunities.

**PEO2**: Embrace new technology to solve real-world problems, whether alone or

As a team, while prioritizing ethics and societal benefits.

**PEO3**: Accept lifelong learning to expand future opportunities in research and

Product development.

**Programme Specific Outcomes (PSOs):**

**PSO1**: Ability to create and use Artificial Intelligence and Machine Learning

Algorithms, including supervised and unsupervised learning, reinforcement

Learning, and deep learning models.

**PSO2**: Ability to collect, pre-process, and analyze large datasets, including data

Cleaning, feature engineering, and data visualization..

**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. **Problemanalysis:**Identify,formulate,reviewresearchliterature,andanalyzecomplex 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 effectivelyon 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** project is a recreation of the classic arcade game, developed using Java programming. The project emphasizes the integration of fundamental and advanced programming concepts to build a fully interactive and engaging application. The game is designed to offer players a seamless experience where they control a snake to consume randomly placed food items on a grid. Each food item consumed increases the snake's length and adds to the player's score. The challenge arises as the snake grows longer, requiring players to skillfully avoid collisions with the walls or the snake's own body.

This project demonstrates the application of core Java concepts such as **Object-Oriented Programming (OOP)**, **graphical user interfaces (GUIs)**, and **event-driven programming**. It leverages the AWT (Abstract Window Toolkit) and Swing libraries for rendering the game interface, while multithreading is used to ensure smooth and responsive gameplay.

The development process follows a modular approach, with distinct components handling tasks such as snake movement, collision detection, scoring, and graphical rendering. The use of a model-view-controller (MVC) architecture ensures a clean separation of concerns, making the codebase easy to understand and extend.

The Snake Game is not only a fun and interactive application but also an excellent platform to practice algorithmic thinking and software engineering principles.

vi

# ABSTRACT WITH POs AND PSOs MAPPING

|  |  |  |
| --- | --- | --- |
| **ABSTRACT** | **POs MAPPED** | **PSOs MAPPED** |
| The Snake Game project focuses on designing and developing a classic arcade game in Java. The game involves controlling a snake to consume food items, which increases its length and score, while avoiding collisions with the walls or its own body. This project applies key Java concepts such as graphical user interfaces (AWT/Swing), event handling, multithreading, and object-oriented programming. The game not only demonstrates problem-solving skills but also reinforces modular software design through clean architecture and structured code. | PO1,PO2,PO3,PO5,PO10 | PSO1,PSO2 |

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

**SUPERVISOR HEAD OF THE DEPARTMENT**

vii

# 

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER** No. | TITLE | **PAGE** No. |
|  | ABSTRACT |  |
| 1 | INTRODUCTION |  |
|  | 1.1 Objective |  |
|  | 1.2 Overview |  |
|  | 1.3 Java Programming concepts |  |
| 2 | PROJECT METHODOLOGY |  |
|  | 2.1 Proposed Work |  |
|  | 2.2 Block Diagram |  |
| 3 | MODULE DESCRIPTION |  |
|  | 3.1 Game Initialization |  |
|  | 3.2 Snake Movement |  |
|  | 3.3 Collision Detection |  |
|  | 3.4 Scoring System |  |
|  | 3.5 GUI Enhancements |  |
| 4 | RESULTS AND DISCUSSION |  |
| 5 | CONCLUSION |  |
|  | REFERENCES |  |
|  | APPENDIX |  |

# viii

# CHAPTER 1

# INTRODUCTION

# Objective

# The goal of the project is to build a fully functional, user-interactive snake game that:

# Enhances programming skills in Java.

# Demonstrates game development using the AWT and Swing libraries.

# Implements object-oriented programming for game state and logic.

# Overview

# The Snake Game involves navigating a snake to eat randomly placed food while avoiding collisions with the walls or its own body. The player’s score increases as the snake eats food, and the game ends when a collision occurs. Key features of the game include:

# Dynamic Gameplay: The snake grows with each food item consumed.

# Keyboard Controls: Use arrow keys for directional movement.

# Game States: Start screen, active gameplay, and game-over screen.

* 1. **Java Programming Concepts**

This project uses several Java concepts:

1. Object-Oriented Programming (OOP): Classes like Snake, Food, and GameBoard encapsulate different functionalities.
2. AWT/Swing Libraries: To build graphical components and render the game.
3. Event Handling: To capture and process keyboard inputs for movement.
4. Multithreading: To manage the game loop for smooth rendering and real-time updates.
5. Collections Framework: To store the snake's body segments and manage dynamic updates**.**

**CHAPTER 2**

**PROJECT METHODOLOGY**

**2.1Proposed Work**

The proposed project aims to design and develop a Snake Game using Java programming. The game will replicate the functionality of the classic Snake Game, incorporating modern programming principles and techniques to deliver a smooth and interactive user experience. The work focuses on using Object-Oriented Programming (OOP), Graphical User Interfaces (GUIs), and event-driven programming. The following steps outline the proposed work in detail:

**Step 1: Conceptualization**

* Define the core objectives of the game:
  + A grid-based game area for the snake to move.
  + A controllable snake that grows as it consumes food.
  + Collision detection for game-over scenarios.
  + A scoring system that tracks the player's progress.
* Finalize tools and technologies: Java (JDK), Swing, AWT, and multithreading.

**Step 2: Design**

* Game Logic Design:
  + Develop algorithms for the snake’s movement, food generation, collision detection, and scoring.
  + Implement boundary checks to ensure the snake stays within the game grid.
* Architecture Design:
  + Follow the Model-View-Controller (MVC) architecture for better modularity and maintainability.
    - Model: Handles the game's data, including the snake’s position, food, score, and game status.
    - View: Manages the graphical representation of the game elements.
    - Controller: Captures user input and updates the game state accordingly.
* Graphical Layout:
  + Plan the visual design of the game, including snake segments, food, and the grid.

**Step 3: Implementation**

* Snake Movement:
  + Create a class to represent the snake with attributes like position, direction, and size.
  + Use a linked list or dynamic array to handle the snake's growing length.
* Food Placement:
  + Implement random food placement logic while ensuring it doesn’t overlap with the snake.
* Collision Detection:
  + Design a system to detect when the snake collides with itself, the game boundaries, or the food.
* User Input Handling:
  + Implement keyboard event handling using Java’s KeyListener to allow the user to control the snake.
* Game Engine:
  + Use a timer or multithreading to manage real-time gameplay updates.
  + Update the game state at regular intervals and redraw the game elements on the screen.

**Step 4: Testing**

* Perform unit testing for each module:
  + Test the snake’s movement and direction changes.
  + Verify food placement and ensure no overlaps with the snake.
  + Check collision detection accuracy.
* Conduct integration testing to ensure seamless interaction between all components.
* Evaluate the game’s performance and responsiveness under various scenarios.

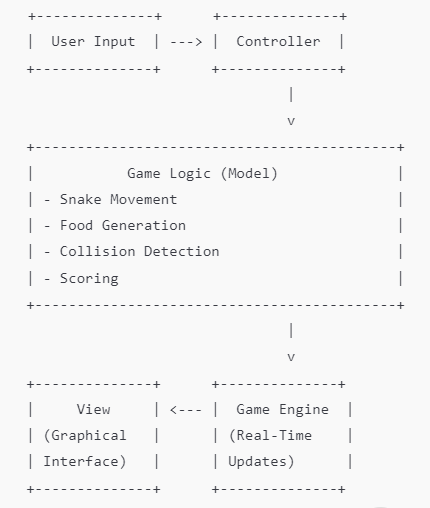
**Step 5: Deployment**

* Package the project into an executable JAR file for easy distribution.
* Provide clear instructions for installation and usage.

**Step 6: Future Enhancements**

* Introduce levels with increasing difficulty.
* Add obstacles within the game grid.
* Include sound effects and animations.
* Explore multiplayer options and mobile compatibility.

**2.2 Block Diagram**

****

**CHAPTER 3**

**MODULE DESCRIPTION**

## 3.1 Module 1 Game Initialization

## This module is responsible for setting up the game environment and initializing all

## the game components.

## Responsibilities:

## Create the game window using a graphical library like Swing or JavaFX.

## Initialize the snake's initial position, direction, and size.

## Generate the first food item randomly on the game grid.

## Set up the game timer to control the game loop.

## Key Features:

## Grid Setup: Establishes the game area where the snake moves.

## Timer: Ensures smooth movement and real-time updates.

## Default Settings: Sets the initial speed, score, and snake size.

## Dependencies:

## Relies on the Snake Module to create the snake and the Food Module to place the first food item.

## 3.2 Module 2 Snake Movement

## This module controls the movement of the snake based on user input and updates its position on the grid.

## Responsibilities:

## Process user input through arrow keys to change the snake’s direction.

## Move the snake smoothly by updating its body’s coordinates based on its current direction.

## Prevent the snake from reversing its direction directly (e.g., moving left when going right).

## Key Features:

## Movement Logic: Updates the position of all segments of the snake.

## Boundary Wrapping (Optional): Allows the snake to reappear on the opposite edge of the grid when hitting the boundary.

## Smooth Transition: Ensures movement without graphical glitches.

## Dependencies:

## Relies on the User Input Module to capture keyboard inputs.

## 3.3 Module 3 Collision Detection

## This module ensures the game handles all possible collisions to maintain the rules of gameplay.

## Responsibilities:

## Detect if the snake collides with:

## Itself: Ends the game if the snake’s head intersects with its body.

## Walls: Ends the game if the snake hits the boundary of the grid.

## Food: Triggers growth and score updates when the snake eats food.

## Key Features:

## Game Over Logic: Displays the game-over screen when a collision occurs.

## Food Collision: Ensures the food is consumed properly and a new food item is generated.

## Dependencies:

## Works with the Snake Module for self-collision detection and the Food Module to handle food interactions.

## 3.4 Module 4  Scoring System

## This module tracks the player's progress and updates the score based on gameplay.

## Responsibilities:

## Increment the score whenever the snake eats food.

## Display the current score dynamically on the game interface.

## Optionally, implement a high-score feature to track the player's best performance.

## Key Features:

## Score Incrementation: Adds points for each food item consumed.

## Real-Time Display: Continuously updates the score on the screen.

## Bonus System (Optional): Adds bonus points for specific achievements like speed or length milestones.

## Dependencies:

## Works with the Food Module to determine when food is eaten.

## 3.5 Module 5 GUI Enhancements

## This module enhances the visual appeal and user interaction of the game.

## Responsibilities:

## Design a visually appealing game window with intuitive controls.

## Use graphics to render the snake, food, grid, and score.

## Display game-over messages and options to restart or exit the game.

## Key Features:

## Game Aesthetics: Improves the look and feel with colors and animations.

## Dynamic Interface: Updates visuals in real time as the game progresses.

## Game Over Screen: Displays a message with the final score and restart options.

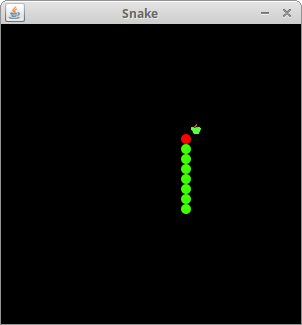
## Pause/Resume Functionality (Optional): Allows the player to pause and resume the game.

## Dependencies:

## Relies on all other modules to provide data for rendering game elements

**CHAPTER 4**

**RESULTS AND DISCUSSION**

****

****

The above screenshots showcases the Java-based Snake Game in action. The game window has a black background, which provides a simple and focused interface for gameplay. The snake is represented by a series of green circles, with the snake's head highlighted in red for easy identification. The food, shown as a small icon (e.g., a fruit), is positioned randomly on the grid for the snake to consume. This classic arcade-style design demonstrates the key mechanics of the game, including real-time movement and food collection, within a visually clean and responsive environment.

**CHAPTER 5**

**CONCLUSION**

The Snake Game project implemented in Java successfully recreated the classic arcade game, highlighting key programming concepts such as object-oriented design, GUI development, and event-driven programming. The game delivered smooth functionality, including snake movement, collision detection, food consumption, and a real-time scoring system, all within an intuitive interface.

Challenges, such as efficient collision handling and food placement, were addressed using logical algorithms and debugging techniques. The project also provided insights into modular design, real-time rendering, and resource optimization, ensuring seamless performance across platforms.

While the current version is functional and engaging, it can be enhanced with features like difficulty levels, multiplayer modes, custom themes, and leaderboards to improve gameplay and replayability. Overall, this project served as both a learning experience and a foundation for exploring more advanced game development concepts in Java.

**REFERENCES:**

**1.Books**:

Java: The Complete Reference by Herbert Schildt.

**2.Websites**:

GeeksforGeeks tutorials on Java Swing.

YouTube channels like "Code With Harry" for Java GUI development.

**3.Documentation**:

Oracle’s official Java documentation.

**APPENDIX**

**(Coding)**

**1.Game Initialization**:

public class GamePanel extends JPanel {

private final int TILE\_SIZE = 25;

private final int GAME\_WIDTH = 500;

private final int GAME\_HEIGHT = 500;

private Timer timer;

public GamePanel() {

setFocusable(true);

setPreferredSize(new Dimension(GAME\_WIDTH, GAME\_HEIGHT));

initializeGame();

}

private void initializeGame() {

// Initialize snake and food positions

// Start the game timer

}

}

**2.Snake Movement:**

public void keyPressed(KeyEvent e) {

switch (e.getKeyCode()) {

case KeyEvent.VK\_UP:

if (direction != DOWN) direction = UP;

break;

case KeyEvent.VK\_DOWN:

if (direction != UP) direction = DOWN;

break;

case KeyEvent.VK\_LEFT:

if (direction != RIGHT) direction = LEFT;

break;

case KeyEvent.VK\_RIGHT:

if (direction != LEFT) direction = RIGHT;

break;

}

}

**3.Collision Detection**:

private boolean checkCollision() {

// Check for wall collision

// Check if snake head intersects with its body

}

**4.Scoring System**:

private void updateScore() {

score++;

repaint(); // Refresh the scoreboard

}