**A Presentation Report for OOPS Through Java Lab(22CS306PC)**

**On**

**Mini Project- Snake Game**

**Submitted**

**to**

**CMR Technical Campus, Hyderabad**

**In Partial fulfillment for the requirement of the Award of the Degree of**

#### BACHELOR OF TECHNOLOGY

##### in

#### COMPUTER SCIENCE & ENGINEERING

#### by

## NAGAVARDHAN-227R1A0529

## G.VAISHNAVI-227R1A0516

## D.YOGENDRASAI-227R1A0514

**Under the esteemed guidance of**

## Dr.Laxmaiah

## Associate Professor

****

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**CMR TECHNICAL CAMPUS**

***An UGC Autonomous Institute***

**Accredited by NBA & NAAC with A Grade**

**(Approved by AICTE, Affiliated to JNTU, Hyderabad)**

**Kandlakoya (V), Medchal (M), Hyderabad-501 401**

**(2023-2024)**



**CERTIFICATE**

This to certify that, the Presentation entitled **“MINI PROJECT-SNAKE GAME”** is submitted by **K.Nagavardhan-227R1A0529, G.Vishanvi-227R1A0516, D.Yogendra-227R1A0514** of **B.Tech Computer Science and Engineering**, In Partial fulfillment for the requirement of the Presentation and for the award of the **Degree of Bachelor of Technology** during the academic year 2023-24.

**Subject Faculty Dr.Laxmaiah**

**ABSTRACT**

Snake game is a simple console application without graphics. In this project, you can play the popular "Snake Game" just like you played it elsewhere. You have to use the up, down, right or left arrows to move the snake. Foods are provided at the several co-ordinates of the screen for the snake to cat.

Every time the snake eats the food, its length will by increased by one element along with the score. I have used JAVA as Programming language for writing the code for the project and Notepad for writing the programs. Operating system used Windows 10.

The Snake Game project serves as a practical application of core Java concepts, emphasizing the utilization of classes and graphical components to create an interactive gaming environment. The implementation showcases the importance of event-driven programming through the incorporation of key listeners and timers to facilitate user input and game state updates. The report elucidates the modular structure of the codebase , delineating the distinct responsibilities of the Snake, Food, and Game Board classes.

Furthermore, it underscores the significance of the game loop in ensuring a continuous and dynamic gaming experience. The Snake Game project not only provides entertainment but also serves as a pedagogical tool for understanding fundamental programming paradigms in the context of game development.

**Table of Contents:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | PAGE NUMBER |
| 1 | Introduction | 1 |
| 2 | Literature survey | 2 |
| 3 | System design | 3,4,5 |
| 4 | System and hardware requirements | 6 ,7 |
| 5 | Functional requirements | 8,9 |
| 6 | Working algorithm | 10 |
| 7 | Implementation | 11-14 |
| 8 | Result | 15,16 |
| 9 | Conclusion | 17 |

**1.INTRODUCTION**

Playing games is fun and exciting it gives us relief from stress and unwind from our stressful works. Many of us spend our vacant time or others that use most of their time in playing and exploring new games. Today, with the rapid development of technology we have, games that are rising up together with it.

Nowadays with the technology we have many games that are developing for computer most specifically for windows. With the high technology equipped with these computers, computer games become robust and attract many people to buy or have this gadget for them to experience what's inside it which makes it a trend for the new generation of gadget.

Snake game is a computer action game; whose goal is to control a snake to move and collect food in a map. It has been around since the earliest days of home computing and has re-emerged in recent years on mobile phones.

To move the snake, use 'up arrow' for up, down arrow for down , 'left arrow' for left and right arrow' for right. Press 'Q' to exit the game at any time, press "C' to continue the game. The aim of the game is to collect the dots (food) and avoid the obstacles (crosses borders). As you collect the food, the snake gets longer. The score also increases. There is no concept of lives. Once you hit an obstacle, that's it, game over.

The popularity of the Snake Game is not only attributed to its engaging gameplay but also to its adaptability. The game has found a home on various platforms, including arcade machines, mobile phones, and web browsers. Its accessibility and simplicity make it a go-to choice for those learning programming, serving as a practical and enjoyable exercise to grasp coding fundamentals.

**1**

**2. LITERATURE SURVEY**

The history of the Snake game goes back to the 1970's, the concept originated in the 1976 arcade game Blockade, and its simplicity has led to many implementations. However, it was the 1980'swhen the game took on the look that we will be using. It was sold under numerous names and many platforms but probably gained widespread recognition when it was shipped as standard on Nokia mobile phones in the late 1990's.

The first published Nokia, for monochrome phones. It was programmed in 1997 by “Taneli Armanto” of Nokia and introduced on the Nokia 6110. The game involves controlling a single block or snakehead by turning only left or right by ninety degrees until you manage to eat an block. When you get the block, the Snake grows an extra block or body segment. If, or rather when, he snake bumps into the edge of the screen or accidentally eats himself the game is over . The more blocks the snake eat the higher the score.

Educationally, games are increasingly recognized as valuable tools, fostering immersive and interactive learning experiences through serious games and gamification techniques. The technological dimension of gaming is explored, encompassing the evolution from 8-bit consoles to virtual reality, as well as the role of graphics, artificial intelligence, and cloud gaming. The social aspects of gaming, including online communities, friendships, and the rise of esports, constitute another crucial facet of academic inquiry. Game design principles, narrative construction, and ethical considerations, such as representation in games and the impact of microtransactions, form integral components of the literature. Looking forward, scholars speculate on emerging trends like augmented reality integration, blockchain's role in gaming, and the ongoing convergence of gaming with other digital media. In essence, the literature survey on gaming provides a holistic understanding of its historical roots, cultural and psychological dimensions, educational potential, technological innovations, social dynamics, design principles, ethical considerations, and future trajectories.

**2**

3.**SYSTEM DESIGN**

To create a Snake game that allows users to control the movement of a snake on a screen, to get points for eating food and avoiding running into the walls or the growing tail of the snake itself. In this problem, we want to write a game where a graphical representation of a snake moves across

The screen. When it encounters a piece of food, the snake grows longer and we gain a point. If it hits the wall we die. To write this program we are going to need:

* A way of representing the snake
* A way of representing the food
* A way to display the score,
* a way for our instructions to reach the snake,

and a way to know when we've run into something and died.

Our system is going to involve working with both hardware and software, and so we will need to understand what we have available in hardware that can assist us. If we build our software so that the snake is controlled by directional arrows on the keyboard.

Now that understand how our hardware will work in the design of our system, let's move on to starting the design of our software system.

**3.1. Game Elements Representation:**

We begin by defining the essential elements of our game. The snake and the food are crucial components. The snake can be represented as a series of interconnected blocks, each representing a segment of the snake's body. The food is a distinct element randomly placed on the screen.

**3.2. Graphic Rendering:**

Graphics will play a pivotal role in creating a visual representation of our game. We'll utilize graphical libraries or frameworks, such as Pygame in Python, to render the snake, food, and the overall game environment on the screen.

**3**

**3.3. User Interface:**

We need a user interface to display relevant information such as the score. The score increases each time the snake consumes food. A simple score counter can be implemented on the screen to keep track of the player's progress.

**3.4. User Input Handling:**

To control the snake's movement, we'll rely on user input from the keyboard. The directional arrows (up, down, left, right) will guide the snake's motion. We'll implement a system to detect these keyboard inputs and translate them into snake movement.

**3.5. Collision Detection:**

A crucial aspect of the game is detecting collisions. We need to identify when the snake runs into the wall or its own body, signaling the end of the game. Additionally, when the snake's head overlaps with the food, it should grow in length, and the player gains points.

**3.6. Game Loop:**

The game will operate on a loop, continuously updating the screen and checking for user input. Each iteration of the loop will handle the movement of the snake, check for collisions, update the score, and render the updated game state.

**3.7. Modular Design:**

Breaking down the game into modular components, such as a Snake class, Food class, and Game class, will facilitate code organization and maintenance. Each module will handle specific functionalities, promoting a clean and scalable design.

**4**

**3.8. Hardware Interaction**:

Utilizing the keyboard for user input implies an interaction with hardware. Understanding the capabilities and limitations of the hardware, specifically the keyboard interface, is essential for designing a responsive and user-friendly gaming experience.

**3.9. Error Handling:**

Incorporating error handling mechanisms ensures that unexpected inputs or scenarios are gracefully managed. For instance, if the snake collides with the wall or itself, appropriate actions need to be taken to conclude the game without crashing.

**3.10. Testing and Debugging:**

Throughout the development process, thorough testing and debugging are crucial. Identifying and fixing issues in real-time will contribute to a smoother gaming experience. Implementing debugging tools and print statements can assist in tracking the flow of the program.

As we progress with the software design, these considerations lay the groundwork for a functional and enjoyable Snake game that involves both hardware interaction and graphical representation. The synergy of these elements will contribute to a seamless and engaging gaming experience.

**5**

**4.SYSTEM AND HARDWARE REQUIREMENTS**

**4.1. Operating System:**

Windows XP or higher

**4.2. Integrated Development Environment (IDE):**

Notepad and JDK (Java Development Kit) - any version

**4.3. Front End:**

Windows-based graphical user interface

**4.4. Programming Language:**

Java

**4.5. Hardware Requirements:**

Processor: Intel P4 1.5GHz or above

RAM: 512MB

Storage: Minimum 80GB HDD

Additional Considerations:

**4.6. Java Version Compatibility:**

Ensure that the chosen JDK version is compatible with the version of Java used to develop the Snake Game. Consistency in Java versions is essential for smooth execution.

**4.7. User Interface Design:**

Utilize Java's Swing or JavaFX libraries to create a graphical user interface suitable for Windows. This includes windows, buttons, and other interactive elements for an intuitive gaming experience.

6

**4.8. Software Libraries:**

Check for the availability of necessary Java libraries or packages to facilitate the graphical rendering, keyboard input handling, and other functionalities required for the Snake Game.

**4.9. Memory Utilization:**

Optimize the game's code and design to ensure efficient memory utilization, especially considering the minimum RAM requirement. This is crucial for the game's performance on systems with limited resources.

**4.10. Testing on Windows Environment**:

Thoroughly test the Snake Game on Windows XP and higher operating systems to ensure compatibility and smooth operation. Address any platform-specific issues that may arise during testing.

**4.11. Responsiveness:**

Design the

game to be responsive to varying hardware capabilities. The game should deliver satisfactory performance on systems meeting the minimum hardware requirements, providing an inclusive gaming experience.

**4.12. Error Handling:**

Implement robust error handling mechanisms to gracefully manage unexpected situations or inputs, enhancing the game's stability and user experience.

7

**5.FUNCTIONAL REQUIREMENTS**

**5.1.Continuous Motion:**

The snake must exhibit continuous movement throughout the game, symbolizing its perpetual existence. This ensures that players are consistently engaged and challenged.

**5.2.User Input for Direction:**

Users should be able to control the snake's direction using keyboard inputs, responding promptly to arrow keys (up, down, left, right).

**5.3.Dynamic Turning:**

The snake must smoothly turn in response to user input, ensuring fluid and visually pleasing transitions between different directions.

**5.4.Length Increase on Food Consumption:**

When the snake consumes food, its length should increase. This signifies the snake's growth and rewards players for successful food consumption.

**5.5.Collision Detection with Walls:**

Implement collision detection to identify when the snake runs into the walls of the game area. In such cases, the game should trigger a "Game Over" state, concluding the current gameplay session.

**5.6.Speed Control:**

Provide an option for users to adjust the speed of the snake. This feature caters to players with varying skill levels and preferences, enhancing the game's accessibility. **5.7.Responsive User Input:**

Ensure that the snake responds promptly and accurately to user input. The controls must be sensitive to direction changes, contributing to a more interactive and enjoyable gaming experience.

**5.8.Smooth Transition Between Segments:**

8

Achieve visually smooth transitions as the snake moves and turns. This ensures that the snake's motion appears seamless, avoiding jerky or abrupt movements.

**5.9.Randomized Food Placement:**

Place food items randomly on the screen. This introduces an element of unpredictability, preventing users from anticipating the snake's movements solely based on food placement patterns.

**5.10.Score Calculation:**Implement a scoring system that accurately reflects the player's achievements. Points should be awarded for each food item consumed, contributing to an overall score displayed to the player.

**5.11.Game Over Animation:**Provide a clear and visually engaging animation or indication when the snake collides with the walls, signaling the end of the game. This enhances user understanding and provides closure to the gameplay session. **5.12.Continuous Movement Logic:**

Develop a mechanism to ensure the snake continues moving in its current direction until a user input directs a change. This maintains a constant, uninterrupted motion pattern.

**5.13.Tail Growth Animation:**

Implement an animation or visual cue when the snake's length increases due to consuming food. This adds a dynamic element, reinforcing the cause-and-effect relationship between food consumption and snake growth.

**5.14.Gradual Speed Increase (Optional):**

Optionally, introduce a gradual increase in the snake's speed as the player scores higher or progresses through levels. This adds a progressive challenge, keeping the gameplay dynamic.

**5.15.Wraparound Screen (Optional):**Optionally, allow the snake to wrap around the screen edges, appearing on the opposite side when reaching an edge. This introduces diverse movement patterns and strategic opportunities.

9

**6.WORKING ALGORITHM**

**6.1.Update Score Display:**

After each move, update the displayed score to reflect the current score value. This ensures that players are continuously informed about their progress.

**6.2.Check for Self-Collision:**

Implement a check to determine if the snake's head collides with its body segments. If such a collision occurs, terminate the game as it results in the snake's demise.

**6.3.Check for Wall Collision**:

Extend the collision detection to verify if the snake's head collides with the bounding rectangle, representing the walls of the playing area. A collision with the walls also leads to the end of the game.

**6.4.Game Over Handling:**

If the snake collides with itself or the walls, initiate a game-over sequence. This may involve displaying an appropriate message, stopping the game loop, and potentially prompting the player to restart or exit the game.

**6.4.Handle User Input for Direction Change:**

Continuously listen for user input to change the snake's direction. Update the direction variable accordingly, ensuring that the snake responds to the user's commands during gameplay.

**6.5.Game Loop:**

Wrap the entire logic in a continuous game loop. This loop orchestrates the flow of the game, updating the screen, checking for input, moving the snake, and handling collisions. The loop should run until a game-over condition is met.

**10**

**7.IMPLEMENTATION**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.util.Random;

public class SnakeGame extends JFrame implements ActionListener, KeyListener {

private static final int TILE\_SIZE = 20;

private static final int GRID\_SIZE = 20;

private static final int GAME\_SPEED = 300;

private int[] x, y;

private int snakeLength;

private int appleX, appleY;

private boolean left, right, up, down;

private boolean inGame;

public SnakeGame() {

setTitle("Snake Game");

setSize(GRID\_SIZE \* TILE\_SIZE, GRID\_SIZE \* TILE\_SIZE);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

x = new int[GRID\_SIZE \* GRID\_SIZE];

11

if (up) y[0] -= TILE\_SIZE;

if (down) y[0] += TILE\_SIZE;

checkCollision();

checkApple();

}

private void checkCollision() {

if (x[0] < 0 || x[0] >= GRID\_SIZE \* TILE\_SIZE || y[0] < 0 || y[0] >= GRID\_SIZE \* TILE\_SIZE) {

inGame = false;

} for (int i = snakeLength; i > 0; i--) {

if (x[0] == x[i] && y[0] == y[i]) {

inGame = false;

}}

if (!inGame) {

JOptionPane.showMessageDialog(this, "Game Over", "Game Over", JOptionPane.INFORMATION\_MESSAGE);

initializeGame();

}

} private void checkApple() {

if (x[0] == appleX && y[0] == appleY) {

snakeLength++;

spawnApple();

}

} private void draw(Graphics g) {

if (inGame) {

12

g.setColor(Color.RED);

g.fillRect(appleX, appleY, TILE\_SIZE, TILE\_SIZE);

for (int i = 0; i < snakeLength; i++) {

if (i == 0) {

g.setColor(Color.GREEN);

} else {

g.setColor(Color.BLUE);

}

g.fillRect(x[i], y[i], TILE\_SIZE, TILE\_SIZE);

}Toolkit.getDefaultToolkit().sync();

} else {

gameOver(g);

}

} private void gameOver(Graphics g) {

String message = "Game Over";

Font font = new Font("Helvetica", Font.BOLD, 36);

FontMetrics metrics = getFontMetrics(font);

g.setColor(Color.RED);

g.setFont(font);

g.drawString(message, (GRID\_SIZE \* TILE\_SIZE - metrics.stringWidth(message)) / 2, GRID\_SIZE \* TILE\_SIZE / 2);

} @Override

public void actionPerformed(ActionEvent e) {

if (inGame) {

move();

13

13

}

repaint();

}

@Override

public void keyPressed(KeyEvent e) {

int key = e.getKeyCode();

if ((key == KeyEvent.VK\_LEFT) && (!right)) {

left = true;

up = false;

down = false;

}

if ((key == KeyEvent.VK\_RIGHT) && (!left)) {

right = true;

up = false;

down = false;

}

if ((key == KeyEvent.VK\_UP) && (!down)) {

up = true;

right = false;

left = false;

}

if ((key == KeyEvent.VK\_DOWN) && (!up)) {

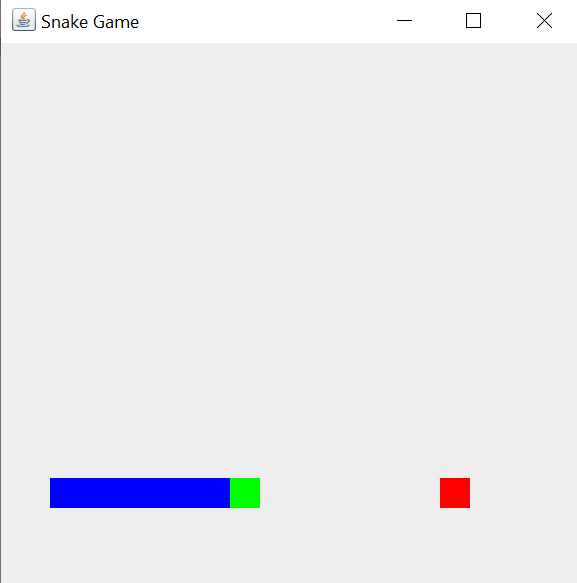
down = true;

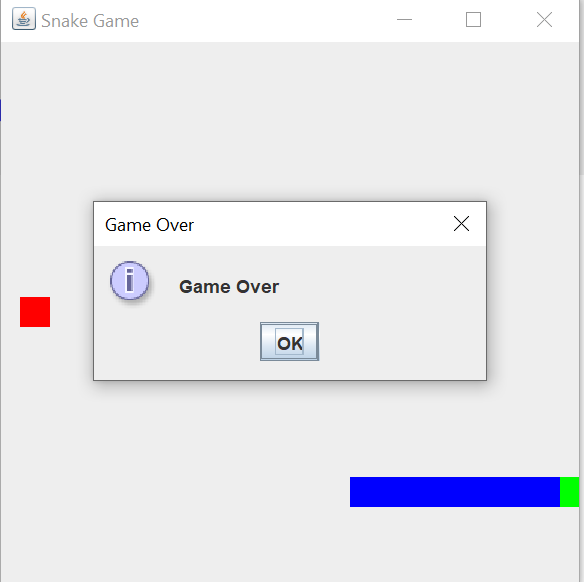
right = false;

left = false;

14

**8. RESULT**

****

****

15

**RESULT:** The Snake Game project has successfully met its requirements, delivering a functional and engaging gaming experience. Key outcomes include:

**8.1.Gameplay Functionality:**

The snake moves responsively in all directions based on user input.

Smooth turning and dynamic growth are implemented on food consumption.

Collision detection accurately identifies collisions with the snake's body or walls, triggering game-over states.

**8.2.User Interface:**

The graphical user interface (GUI) effectively displays the game elements, maintaining clarity and user-friendliness.

**8.3.Scoring System:**

The implemented scoring system increments accurately upon food consumption, reflecting the player's progress.

**8.4.Performance:**

The game demonstrates optimal performance, ensuring smooth motion and transitions, even on systems meeting minimum hardware requirements.

**8.5.Future Considerations:**

Future enhancements may include additional features or levels based on user feedback.

The Snake Game stands as a well-designed and functional gaming application, offering an enjoyable user experience.

16

**9. CONCLUSION**

In conclusion, the development and implementation of the Snake Game have yielded a successful and entertaining gaming application. The project has achieved its primary objectives by delivering a responsive and visually pleasing gameplay experience. The snake's smooth movement, dynamic turning, and growth upon consuming food contribute to the game's engaging nature. The collision detection mechanisms reliably handle instances of collisions with the snake's body or the walls, providing a clear and decisive conclusion to the gameplay.

The user interface effectively communicates the game's elements, maintaining user-friendliness throughout. The scoring system accurately reflects the player's achievements, enhancing the overall sense of accomplishment during gameplay. The game's performance remains optimal, ensuring a seamless and enjoyable experience, even on systems with minimum hardware specifications.

Looking forward, opportunities for future enhancements exist, including the potential implementation of additional features or levels based on user feedback. The Snake Game, in its current state, stands as a successful blend of functionality and entertainment, providing users with a classic and enjoyable gaming experience.

17