Report On

Snake Game

Submitted in partial fulfillment of the requirements of the Course project in Semester III of Second Year Artificial Intelligence and Data Science

by
Viraj Oza
Sarvesh Surve
Konisha Thakare
Maitree Pimple

Supervisor
Prof. Sneha Yadav

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Artificial Intelligence and Data Science

(2023-24)

Vidyavardhini's College of Engineering & Technology
Department of Artificial Intelligence and Data Science

CERTIFICATE

This is to certify that the project entitled “Snake Game” is a bonafide work of " Konisha Thakare, Maitree Pimple, Viraj Oza, Sarvesh Surve" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester III of Second Year Artificial Intelligence and Data Science engineering.

Supervisor

Prof. Sneha Yadav

Dr. Tatwadarshi P. N.
Head of Department

Abstract

A timeless arcade-style video game, The Snake Game is renowned for its straightforward yet addicting gameplay. We explore the creation of a Java-based Snake Game in this microproject. The process of making a useful and interesting game in order to give them a hands-on experience, especially for those who are new to programming.

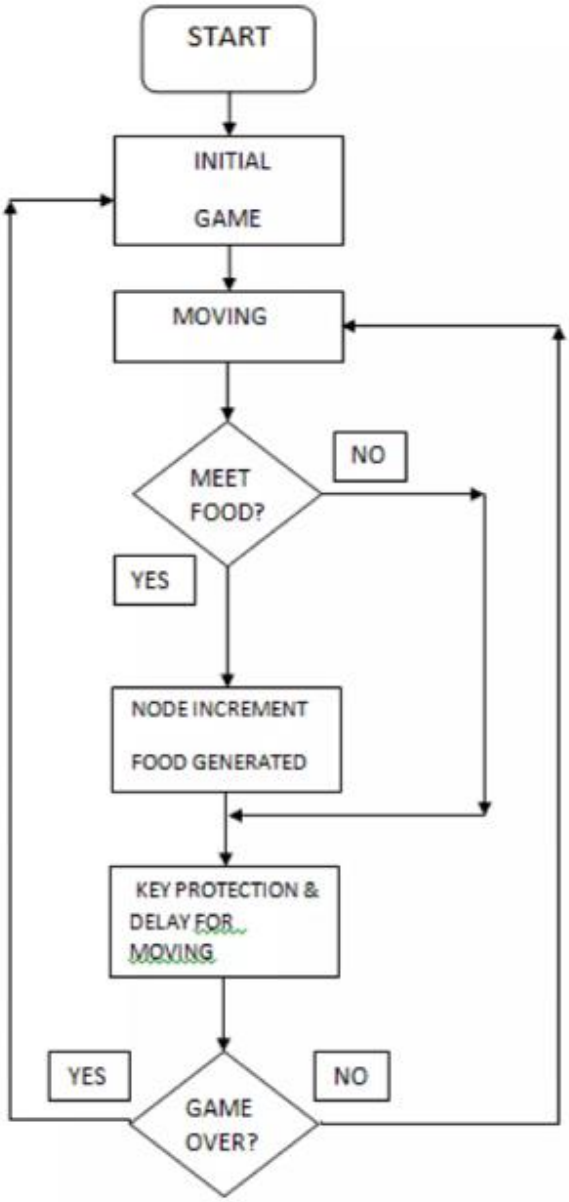
The Snake Game is a great option for teaching because of its historical relevance as a beloved and well-known game and its ability to reinforce basic programming ideas. This microproject report functions as a thorough manual, outlining the technologies employed, the organisation of the project, the use of code, and the difficulties encountered throughout development. It also provides information on the testing procedures, possible future developments, and user interface design of the game.

Table of Contents

Problem Statement:

Develop a fun, engaging, and educational Java-based Snake Game that teaches beginners to programming basic concepts such as loops, variables, and functions.

Block Diagram:



Module Description

A snake game is a simple arcade game in which a player controls a snake that must move around a board while avoiding obstacles and eating food. The snake grows longer each time it eats food, and the game ends if the snake collides with itself or the walls of the board.

The following is a module description of a snake game in Java:

Game Logic Module:

This module is responsible for the core gameplay logic of the game, such as:

- Maintaining the game state, including the position of the snake, the position of the food, and the score.
- Moving the snake in response to player input.
- Checking for collisions between the snake and itself, the walls of the board, and the food.
- Generating new food when the snake eats the current food.
- Determining whether the game is over.

Input Handler Module:

This module is responsible for handling player input and translating it into game commands. For example, when the player presses the up arrow key, the input handler module would send a command to the game logic module to move the snake up.

View Module:

This module is responsible for rendering the game state to the screen. This includes drawing the snake, the food, the score, and any other game elements.

The three modules interact with each other as follows:

1. The game logic module updates the game state based on player input and the current game state.
2. The input handler module handles player input and translates it into game commands.
3. The view module renders the game state to the screen.

The game loop repeats these steps until the game is over.

In addition to the three core modules, a snake game may also include other modules, such as a:

- **Sound Module:** This module would be responsible for playing sound effects and music during the game.
- **High Score Module:** This module would be responsible for tracking the player's high score and displaying it to the player.
- **Menu Module:** This module would be responsible for displaying the game's menu and handling user input on the menu.

These additional modules are not essential for the core gameplay of the game, but they can make the game more enjoyable and engaging for the player.

Brief description

The software and hardware used to implement a snake game in Java using Visual Studio Code can be summarized as follows:

Software:

- Visual Studio Code
- Java Development Kit (JDK)

Hardware:

- A computer with a Java Virtual Machine (JVM) installed

Programming:

The following steps can be taken to program a snake game in Java using Visual Studio Code:

- Create a new Java project in Visual Studio Code.
- Create the following classes:
    - **SnakePanel:** This class should extend JPanel and represent the snake game canvas. It should have methods to draw the snake, food, and other game elements.
    - **SnakeFrame:** This class should extend JFrame and contain the SnakePanel. It should also have methods to start and stop the game loop.
    - **Main:** This class should contain the main() method, which will start the game.
    -
- Write the code for each class.
- Create a main() method to start the game loop. The game loop should repeatedly update the game state, render the game state to the screen, and handle player input.

Code:

JavaFrame.java

```
import javax.swing.*;

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

}
```

JavaPanel.java

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

public class GamePanel extends JPanel implements ActionListener {
    static final int SCREEN_WIDTH = 600;
    static final int SCREEN_HEIGHT = 600;
    static final int UNIT_SIZE = 15;    // size of squares
```

```
    static final int GAME_UNITS =
(SCREEN_WIDTH*SCREEN_HEIGHT)/UNIT_SIZE;
    static final int DELAY = 75;        // speed
    final int x[] = new int[GAME_UNITS];
    final int y[] = new int[GAME_UNITS];
    int bodyParts = 2;
    int applesEaten = 0;
    int appleX;
    int appleY;
    char direction = 'R';   // start direction
    boolean running = false;
    Timer timer;
    Random random;
    boolean gameOverScreen = false;
    JButton restartButton;
    JButton exitButton;
    int selectedButtonIndex = 0;

    GamePanel(){
        random = new Random();
        Dimension dimention = new Dimension(SCREEN_WIDTH, SCREEN_HEIGHT);
        this.setPreferredSize(dimention);
        this.setBackground(new Color(48, 48, 48));
        this.setFocusable(true);
        this.setLayout(null);       // for custom component positioning
        MyKeyAdapter myKeyAdapter = new MyKeyAdapter();
        this.addKeyListener(myKeyAdapter);
        startGame();
    }
```

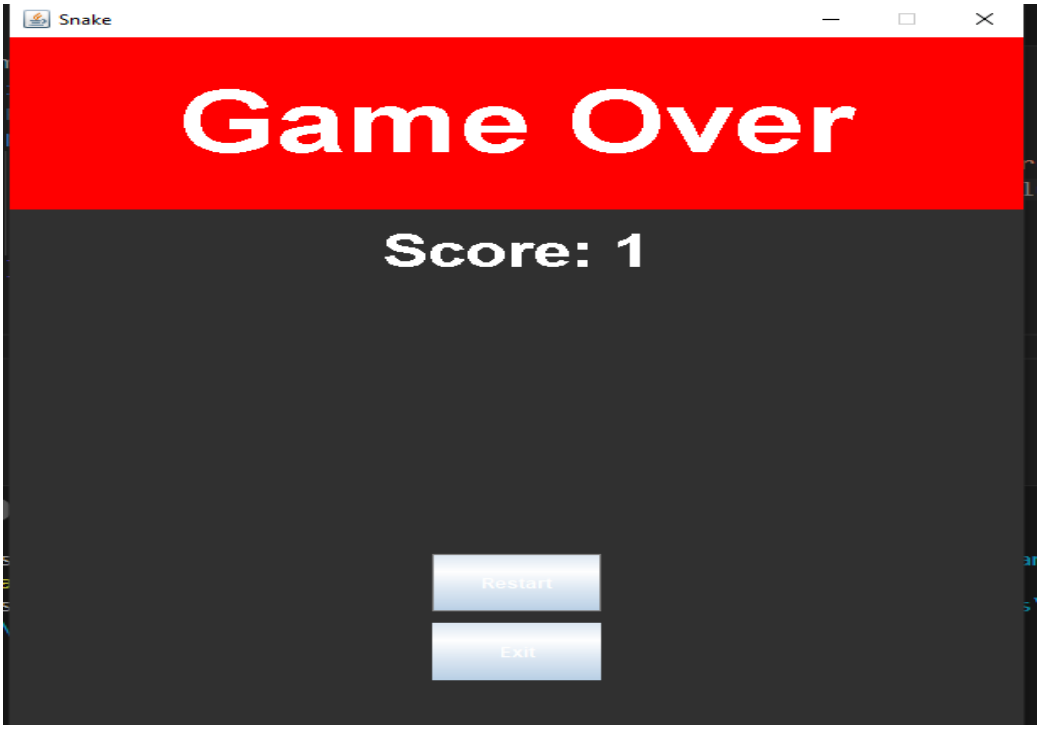
```
public void newApple(){
    appleX = random.nextInt((int)(SCREEN_WIDTH/UNIT_SIZE))*UNIT_SIZE;
    appleY = random.nextInt((int)(SCREEN_HEIGHT/UNIT_SIZE))*UNIT_SIZE;
}

public 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] = y[0] - UNIT_SIZE;
            break;
        case 'D':
            y[0] = y[0] + UNIT_SIZE;
            break;
        case 'L':
            x[0] = x[0] - UNIT_SIZE;
            break;
        case 'R':
            x[0] = x[0] + UNIT_SIZE;
            break;
    }
}

public void checkCollisions(){
    // checks if head collides with body
    for (int i = bodyParts; i>0; i--){
        if ((x[0] == x[i]) && (y[0] == y[i])){
            running = false;
        }
```

```
    }
    // checks if head touches boarders
    if (x[0] < 0){
        running = false;
    }
    if (x[0] > SCREEN_WIDTH){
        running = false;
    }
    if (y[0] < 0){
        running = false;
    }
    if (y[0] > SCREEN_HEIGHT){
        running = false;
    }

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

}

public void checkApple(){
    if ((x[0] == appleX) && (y[0] == appleY)){
        bodyParts++;
        applesEaten++;
        newApple();
    }
}

public void gameOver(Graphics g){
    // show game over text
    g.setColor(Color.red);
    g.fillRect(0, 0, SCREEN_WIDTH, 150); // Position the banner at the to
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        System.setProperty("apple.laf.useScreenMenuBar", "true");
        System.setProperty("apple.awt.application.appearance", "system");

        GameFrame frame = new GameFrame();
    }
}
```

Output:



References:

https://docs.oracle.com/javase/8/docs/api/java/util/Random.html
https://docs.oracle.com/javase/tutorial/uiswing/events/
https://www.geeksforgeeks.org/design-snake-game/