# SNAKE GAME USING JAVA



An

Object-Oriented Programming through Java Course Project Report

in partial fulfilment of the degree

## Bachelor of Technology

In

## Computer Science & Engineering

**By**

K. Vidhya Reddy 2103A51518

B. Shiva Sathvika 2103A51550

P. Aishwarya Reddy 2103A51103

Submitted to **G. Ranjith** Sir





# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

**CERTIFICATE**

This is to certify that the Object **Oriented Programming through Java** - Course Project Report entitled “ SNAKE GAME USING JAVA ” is a record of bonafide work carried out by students K . Vidhya Reddy, B.Shiva Sathvika, P. Aishwarya Reddy bearing Roll No(s) 2103A51518, 2103A51550, 2103A51103 during the academic year 2023-2024 in partial fulfilment of the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering** by SR University, Anantsagar.

**Lab In-charge Head of the Department**

# 

# TABLE OF CONTENTS

1. ABSTRACT
2. OBJECTIVE OF THE PROJECT
3. DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT
4. DESIGN
   1. SCREENS
5. IMPLEMENTATION
   1. CODE
6. RESULT SCREENS
7. CONCLUSION

## ABSTRACT

The Snake game is a classic arcade game where the player controls a snake that moves

around the screen, eating food and growing longer. The objective is to avoid colliding with the walls or the snake's own body. The game ends when the snake crashes into a wall or itself. The project will involve implementing the game logic, handling user input, and displaying the game on the screen

# OBJECTIVE OF THE PROJECT

The Project entitled **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.

The Snake game is a classic arcade game where the player controls a snake that moves around the screen, eating food and growing longer. The objective is to avoid colliding with the walls or the snake's own body. The game ends when the snake crashes into a wall or itself. The project will involve implementing the game logic, handling user input, and displaying the game on the screen.

# DEFINITIONS OF THE ELEMENTS USED IN THE PROJECT

**Player Control**:

The snake's movement is under the player's control, typically using keyboard input (arrow keys or other designated keys).

**Growth**:

When the snake consumes food, it grows longer by adding new segments to its tail. The challenge forthe player is to strategically maneuver the snake to consume food and grow as long as possible.

|  |  |
| --- | --- |
| **Consumption** | : When the snake's head collides with a food item, the food is "eaten," removed from |

the game board, and the snake grows longer.

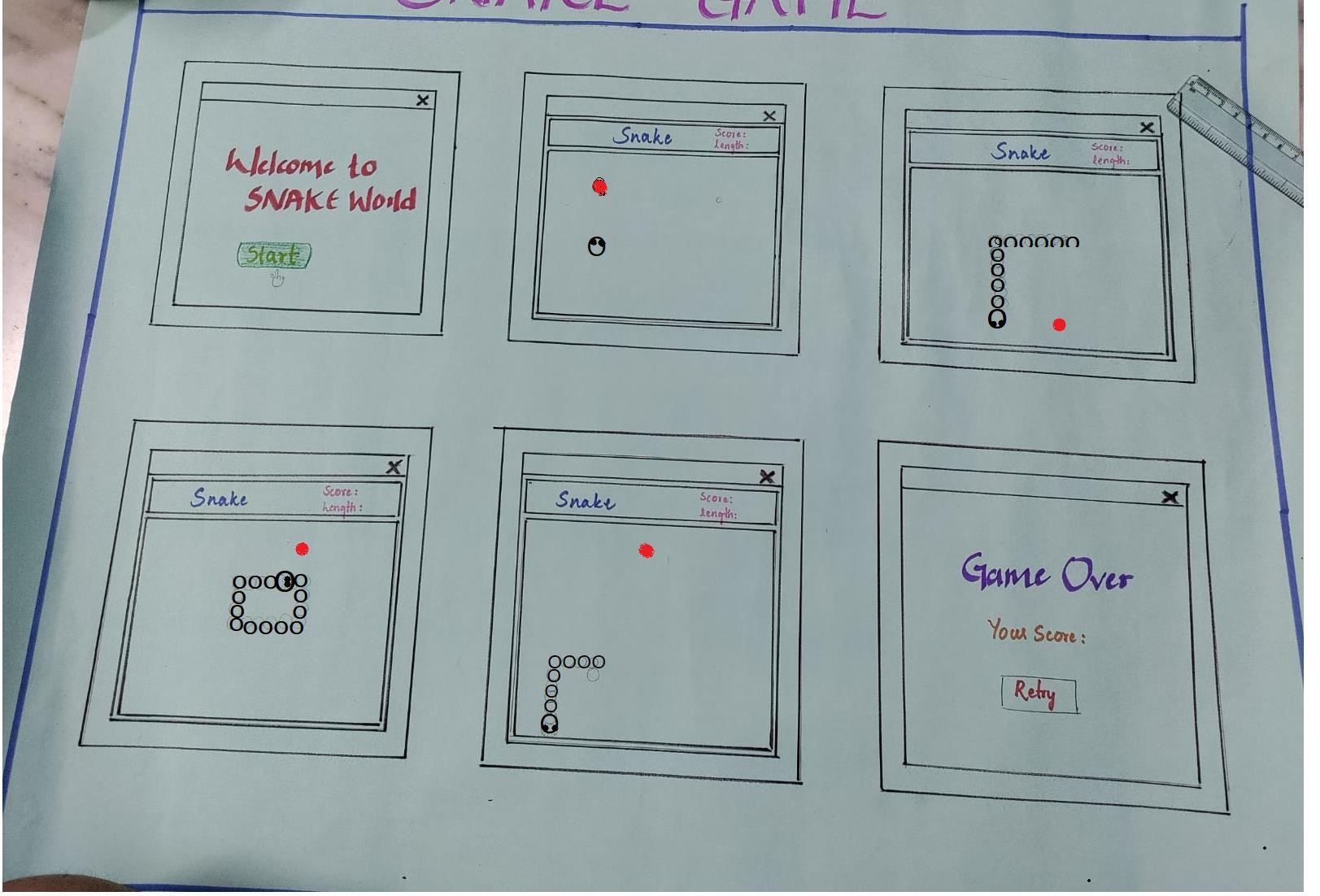
|  |  |
| --- | --- |
| **Scoring** | : Eating food increases the player's score, making it a key aspect of the game's objective. |

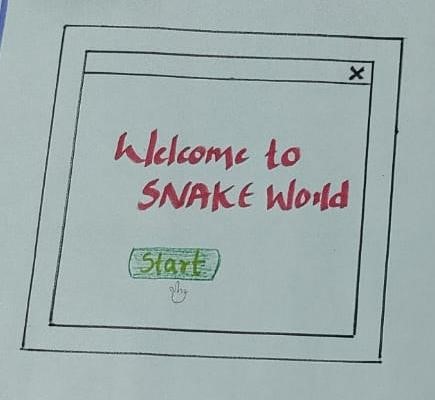
|  |  |
| --- | --- |
| **Trigger** | : This state is triggered when specific conditions are met, such as the snake colliding with |

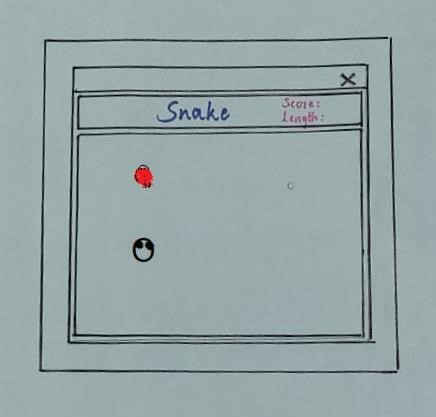
walls or itself.

**4.DESIGN :**

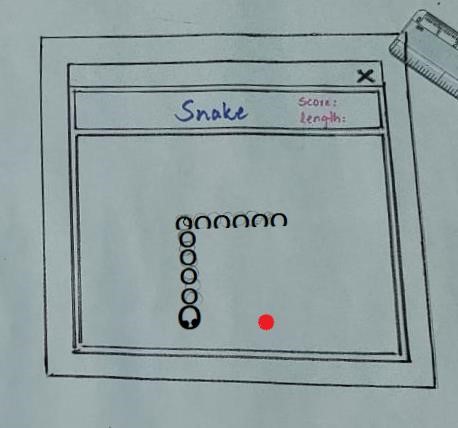
**4.1.SCREENS :**

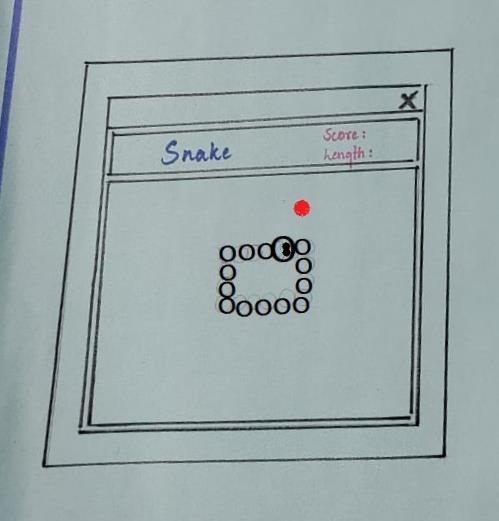


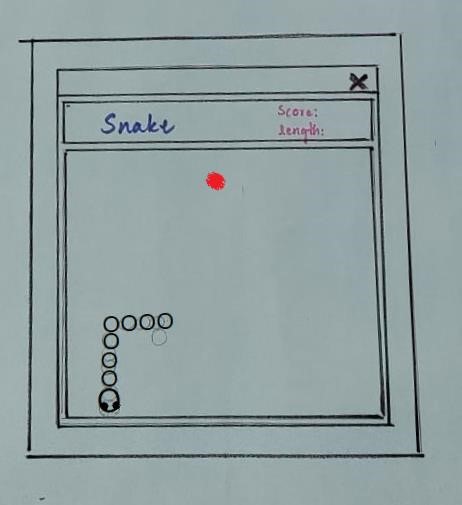


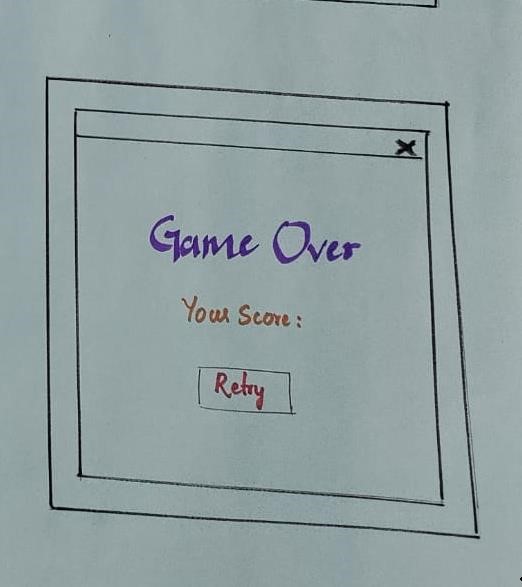


**7**









## 5.IMPLEMENTATION

**5.1.CODE**

**LoginPage.java :**

import java.awt.EventQueue; import javax.swing.JFrame; import javax.swing.JLabel; import java.awt.Font; import javax.swing.JTextField; import javax.swing.JPasswordField; import javax.swing.JButton; import javax.swing.JOptionPane;

public class LoginPage {

private JFrame frame;

private JTextFieldtextField;

private JPasswordFieldpasswordField;

public static void main(String[] args) { EventQueue.invokeLater(new Runnable() { public void run() {

try {

LoginPage window = new LoginPage(); window.frame.setVisible(true);

}

catch (Exception e) { e.printStackTrace();

}

}

});

}

public LoginPage() {

initialize();

}

private void initialize() {

frame = new JFrame();

frame.setBounds(100, 100, 450, 300); frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.getContentPane().setLayout(null);

JLabellblNewLabel = new JLabel("USERNAME :"); lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD | Font.ITALIC, 14)); lblNewLabel.setBounds(127, 30, 133, 23); frame.getContentPane().add(lblNewLabel);

textField = new JTextField();

textField.setBounds(127, 70, 133, 30); frame.getContentPane().add(textField); textField.setColumns(10);

JLabel lblNewLabel\_1 = new JLabel("PASSWORD :"); lblNewLabel\_1.setFont(new Font("Times New Roman", Font.BOLD | Font.ITALIC, 14)); lblNewLabel\_1.setBounds(127, 111, 97, 23); frame.getContentPane().add(lblNewLabel\_1);

JButtonbtnNewButton = new JButton("Login"); btnNewButton.setFont(new Font("Times New Roman", Font.BOLD | Font.ITALIC, 14)); btnNewButton.setBounds(145, 202, 97, 23); frame.getContentPane().add(btnNewButton);

passwordField = new JPasswordField();

passwordField.setBounds(127, 145, 133, 30); frame.getContentPane().add(passwordField);

btnNewButton.addActionListener(e -> {

String username = textField.getText();

char[] password = passwordField.getPassword(); if (isValidCredentials(username, password)) {

frame.dispose();

openPasswordManager();

}

else {

JOptionPane.showMessageDialog(frame, "Login failed.

Please check your credentials.", "Error",

JOptionPane.ERROR\_MESSAGE);

}

});

}

private booleanisValidCredentials(String username, char[] password) {

String validUsername = "JAVAPRO"; String validPassword = "Password";

return username.equals(validUsername) && new

String(password).equals(validPassword);

}

// Open the PasswordManager application private void openPasswordManager() { PasswordManager.main(new String[]{});

}

}

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

import javax.swing.JPanel;

public class GamePanel extends JPanel implements ActionListener{

private static final long serialVersionUID = 1L;

static final int WIDTH = 500;

static final int HEIGHT = 500;

static final int UNIT\_SIZE = 20;

static final int NUMBER\_OF\_UNITS = (WIDTH \* HEIGHT) / (UNIT\_SIZE \* UNIT\_SIZE);

// hold x and y coordinates for body parts of the snake final int x[] = new int[NUMBER\_OF\_UNITS];

final int y[] = new int[NUMBER\_OF\_UNITS];

// initial length of the snake

int length = 5;

intfoodEaten;

intfoodX;

intfoodY;

char direction = 'D';

boolean running = false;

Random random;

Timer timer;

GamePanel() {

random = new Random();

this.setPreferredSize(new Dimension(WIDTH, HEIGHT));

this.setBackground(Color.DARK\_GRAY);

this.setFocusable(true);

this.addKeyListener(new MyKeyAdapter());

play();

}

public void play() {

addFood();

running = true;

timer = new Timer(80, this);

timer.start();

}

@Override

public void paintComponent(Graphics graphics) { super.paintComponent(graphics);

draw(graphics);

}

public void move() {

for (inti = length; i> 0; i--)

{

// shift the snake one unit to the desired direction to create a move x[i] = x[i-1];

y[i] = y[i-1];

}

if (direction == 'L')

{

x[0] = x[0] - UNIT\_SIZE;

}

else if (direction == 'R')

{

x[0] = x[0] + UNIT\_SIZE;

}

else if (direction == 'U')

{

y[0] = y[0] - UNIT\_SIZE;

}

Else

{

y[0] = y[0] + UNIT\_SIZE;

}

}

public void checkFood() {

if(x[0] == foodX&& y[0] == foodY) {

length++;

foodEaten++;

addFood();

}

}

public void draw(Graphics graphics)

{

if (running) {

graphics.setColor(new Color(210, 115, 90)); graphics.fillOval(foodX, foodY, UNIT\_SIZE, UNIT\_SIZE);

graphics.setColor(Color.white); graphics.fillRect(x[0], y[0], UNIT\_SIZE, UNIT\_SIZE);

for (inti = 1; i< length; i++) {

graphics.setColor(new Color(40, 200, 150)); graphics.fillRect(x[i], y[i], UNIT\_SIZE, UNIT\_SIZE);

}

graphics.setColor(Color.white);

graphics.setFont(new Font("Sans serif", Font.ROMAN\_BASELINE, 25)); FontMetrics metrics = getFontMetrics(graphics.getFont()); graphics.drawString

("Score: " + foodEaten, (WIDTH - metrics.stringWidth("Score: " + foodEaten)) / 2, graphics.getFont().getSize());

} else

{ gameOver(graphics);

}

}

public void addFood() {

foodX = random.nextInt((int)(WIDTH / UNIT\_SIZE))\*UNIT\_SIZE; foodY = random.nextInt((int)(HEIGHT / UNIT\_SIZE))\*UNIT\_SIZE;

}

public void checkHit() {

// check if head run into its body

for (inti = length; i> 0; i--)

{

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

{

running = false;

}

}

// check if head run into walls

if (x[0] < 0 || x[0] > WIDTH || y[0] < 0 || y[0] > HEIGHT)

{

running = false;

}

if(!running) {

timer.stop();

}

}

public void gameOver(Graphics graphics) { graphics.setColor(Color.red);

graphics.setFont(new Font("Sans serif", Font.ROMAN\_BASELINE, 50));

FontMetrics metrics = getFontMetrics(graphics.getFont());

graphics.drawString("Game Over", (WIDTH - metrics.stringWidth("Game Over")) / 2, HEIGHT / 2)

graphics.setColor(Color.white);

graphics.setFont(new Font("Sans serif", Font.ROMAN\_BASELINE, 25));

metrics = getFontMetrics(graphics.getFont());

graphics.drawString

("Score: " + foodEaten, (WIDTH - metrics.stringWidth("Score: " + foodEaten)) / 2, graphics.getFont().getSize());

}

@Override

public void actionPerformed(ActionEvent arg0) {

if (running) {

move();

checkFood(); checkHit();

}

repaint();

}

public class MyKeyAdapter extends KeyAdapter {

@Override

public void keyPressed(KeyEvent e) { switch(e.getKeyCode()) {

case KeyEvent.VK\_LEFT:

if (direction != 'R') {

direction = 'L';

}

break;

case KeyEvent.VK\_RIGHT:

if (direction != 'L') {

direction = 'R';

}

break;

case KeyEvent.VK\_UP: if (direction != 'D') {

direction = 'U';

}

break;

case KeyEvent.VK\_DOWN: if (direction != 'U') {

direction = 'D';

}

break;

}

}

}

}

public class SnakeGame {

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

}

}

import javax.swing.JFrame;

public class GameFrame extends JFrame{

private static final long serialVersionUID = 1L;

GameFrame() {

GamePanel panel = new GamePanel();

this.add(panel);

this.setTitle("snake"); this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); this.setResizable(false);

this.pack();

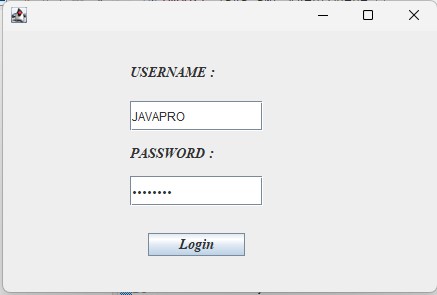
this.setVisible(true);

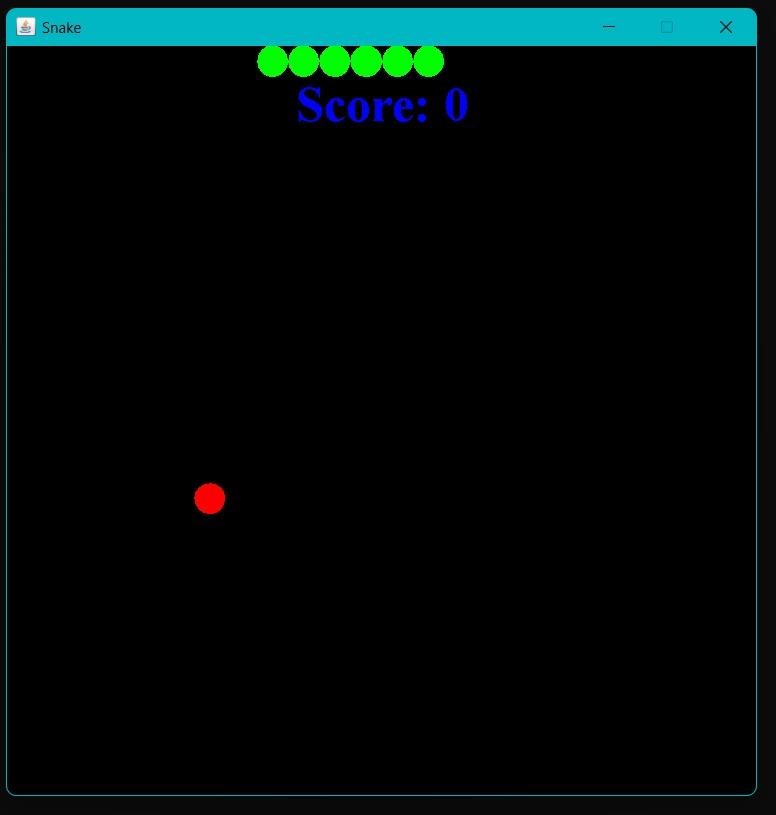
this.setLocationRelativeTo(null);

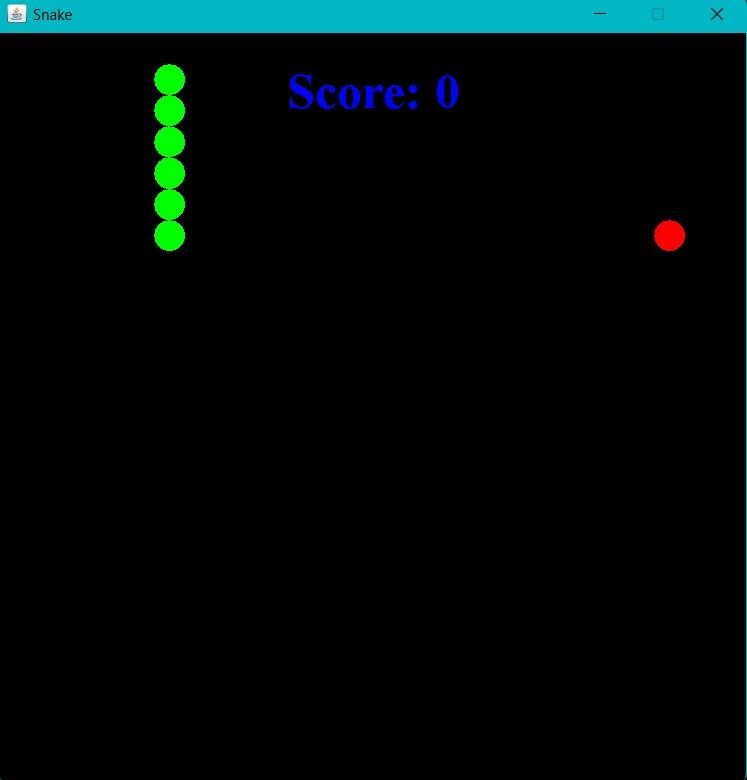
}

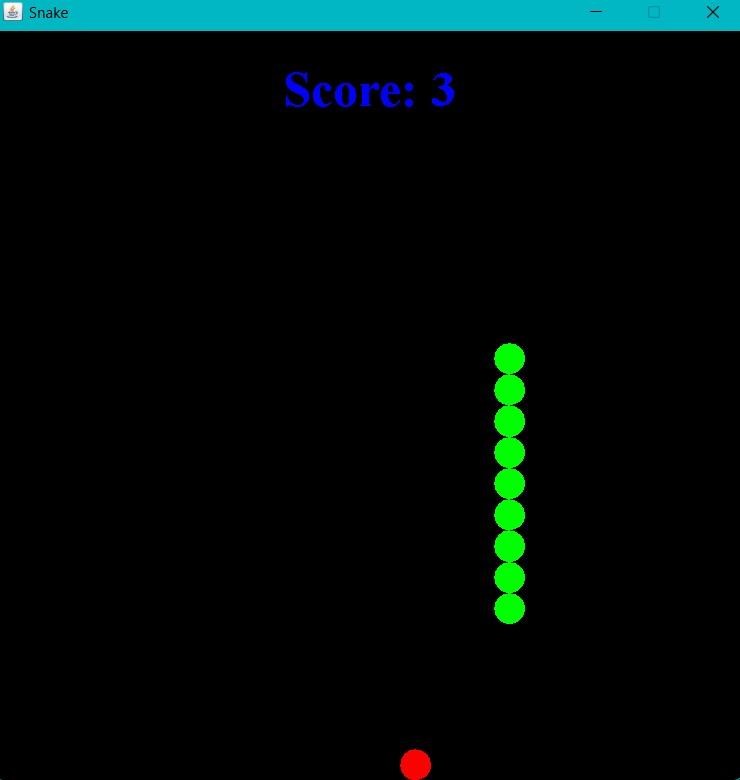
}

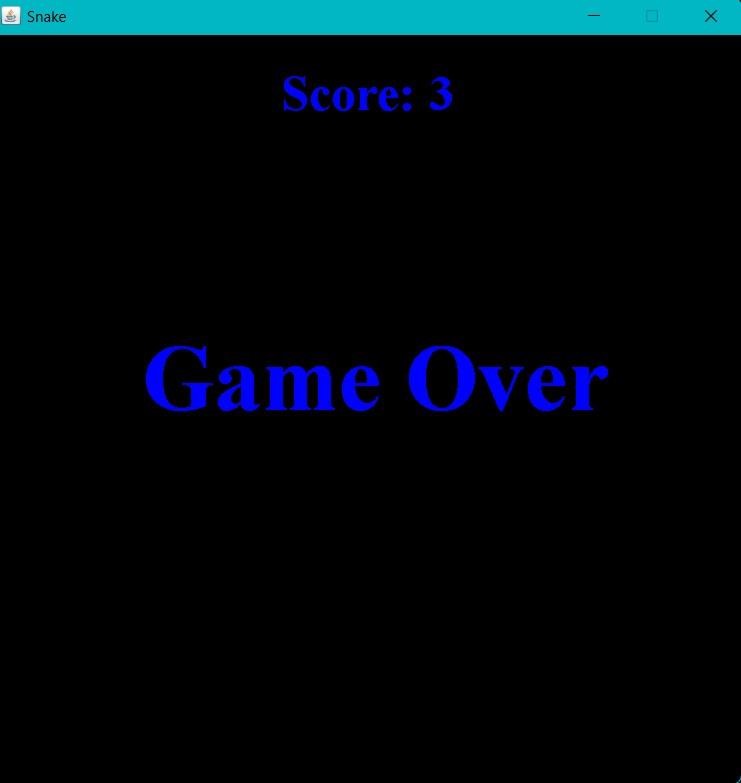
**6.RESULT SCREENS :**











**7.CONCLUSION :**

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 coordinates of the screen for the snake to eat. Every time the snake eats the food, its length will increased by one element along with the score.It isn’t the world’s greatest game, but it does give you an idea of what you can achieve with a relatively simple python programming,and perhaps the basis by which to extend the principles and create 2 interesting games on your own .