Ex.No:11 MINI PROJECT

TABLE OF CONTENTS:

|  |  |  |
| --- | --- | --- |
| S.No | CONTENTS | Page.No |
| 1. | Project Description | 02 |
| 2. | Algorithm and Flowchart | 08 |
| 3. | Source Code | 11 |
| 4. | Actual Output | 17 |
| 5. | Result | 19 |

**1.Project Description:**

* Introduction about the project.
* Scope of the project.
* Hardware /Software requirements.

i.Introduction about the project:

SNAKE (Video Game Genre)

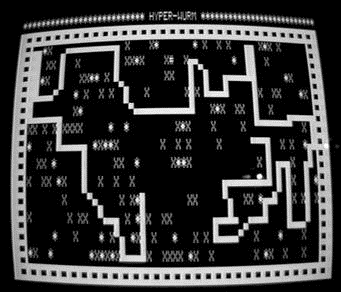


Figure 1:Early Stage of the Snake Game



Figure 2:Snake game in Nokia 3310

Snake is a video game genre where the player maneuvers a growing line that becomes a primary obstacle to itself. The concept originated in the 1976 two-player Arcade game Blockade from Gremlin Industries, and the ease of implementation has led to hundreds of versions (some of which have the word snake or worm in the title) for many platforms ,as shown in Figure 1 . 1982's Tron arcade game, based on the film, includes snake gameplay for the single-player Light Cycles segment. After a variant was preloaded on Nokia mobile phones in 1998, there was a resurgence of interest in snake games as it found a larger audience, as shown in Figure 2.

It was first created in late 70s. Later it was brought to PCs. In this game the player controls a snake. The objective is to eat as many apples as possible. Each time the snake eats an apple its body grows. The snake must avoid the walls and its own body. This game is sometimes called “Nibbles” .

In simple context ,The player controls a long, thin creature, resembling a snake, which roams around on a bordered plane, picking up food (or some other item), trying to avoid hitting its own tail or the edges of the playing area. Each time the snake eats a piece of food, its tail grows longer, making the game increasingly difficult. The user controls the direction of the snake's head (up, down, left, or right), and the snake's body follows. The example on how a prototype model of a snake can be seen in Figure 3.

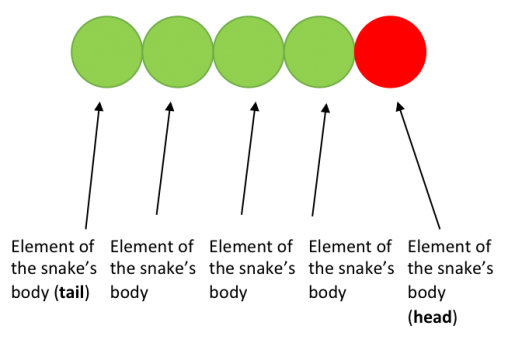


Figure 3:body part of a prototype model of a snake

In 1996,Next Generation ranked it number 41 on their "Top 100 Games of All Time", citing the need for both quick reactions and forethought. In lieu of a title they listed it as "Snake game" in quotes.

On November 29 2012, the Museum of to Modern Art in New York City announced that the Nokia port of Snake was one of 40 games that the curators wished to add to the museum's collection in the future

This 2022 , **Snake Game model** was created using **Java programming language** with the help of **graphical user interface(GUI).** The working of this project is very easy, like the command given in this game is simply the 4 arrow keys and the rules of this project are also simple .

1.Don’t make the snake to eat its own tail.

2.Don’t hit the borders of the screen.

ii.Scope of the project:

* Mission Statement.
* Construction Plan.
* Scope.
* Resource .
* Project execution and Approach.

Mission Statement:

The ultimate goal is to make a snake game with the help of java programming language in addition to GUI

Construction plan:

The plan was pointed out as follows ,

* To know which IDE that I’m going to work with.
* After that , want to create a new project.
* In that project, create 3 classes (**SnakeGame**, **GamePanel, GameFrame**).
* Let **GameFrame** classe gets connected with the main class using the concept of inheritance.
* In **GamePanel** class gets connected with **GameFrame** class using the concept of inheritance.
* Declare the all the things which we want to display on the screen in **GameFrame** class.
* Declare all the required methods and functions to be implemented in the project in **GamePanel** class .
* After completing all the necessary coding part , check the coding part once again , whether every line of the project are perfectly written.
* After writing the codes , don’t forget to add comment lines , which will make the non-programmers to understand what goes in that particular block of lines.
* After completing ,run the program and check whether we got the output correctly.

Scope(lifetime) :

The scope of the project is until the end part of the program ,which displayed on the window as “**GAME OVER**” as well as with the “**Score**” of the game .

Resource:

Actually most part of this project was developed by me ,but the key idea of this project’s program part was all given by an you-tuber . As well as some Wikipedia and other websites links are as follows.

* Youtuber name: **Bro code**.
* Bro Code Youtube : <https://youtube.com/c/BroCodez>.
* Wikipedia : <https://en.wikipedia.org/wiki/Snake_(video_game_genre)>
* Zet Code : <https://zetcode.com/javagames/snake/>
* Learn Java Coding: <https://www.learnjavacoding.com/definitions/snake/>

Project execution and Approach:

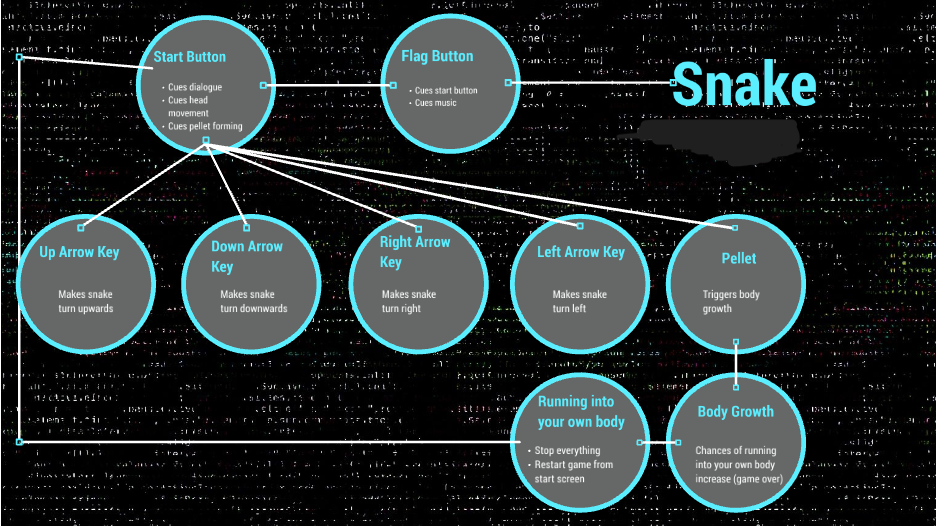


Figure 4:flow of execution

The flow of execution of this game can be clearly seen in the above image Figure 4. The key abstract of the process in the above image is ,there are about 4 arrow keys are used and a pellet or an apple is made to appear in random and when the snake consumes the pellet, the body of the snake should grow and when the snake is unfortunately made to bite its tail or made to hit against the wall, automatically the game comes to an END.

Alright, the first thing you have to do is to split the assignment into 3 parts. You have of course the main game, but you also have 2 other classes. To sum it over here:

* Main
* GameFrame
* GamePanel

**Main Structure:**

Ok, now what are we going to see is to , how to approach the concept

* check methods
* changing directions
* set methods (for example : draw(), move(), etc,.).
* game over method

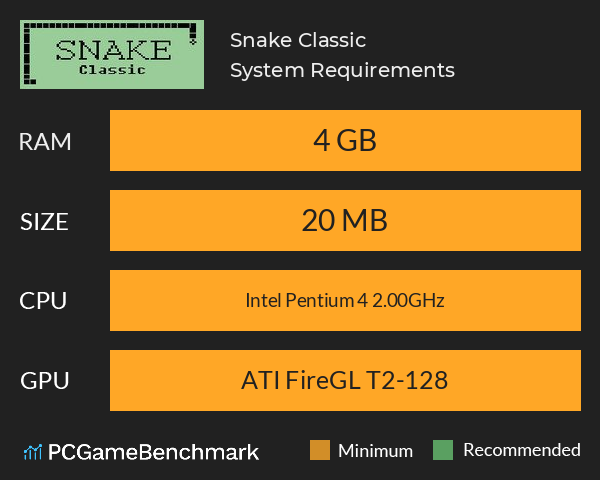
A further breakdown is more precise:

* check methods
  + check boundaries/walls
  + check apple
* changing directions
  + up
  + down
  + left
  + right
* set methods
  + set apple
  + set snake start position(default as right).
* game over method ==> when the snake hits itself (or) slams itself against the wall.

**iii.Hardware/Software requirements:**

* **Hardware requirements:**

Snake game requires a state that you will need at least 256 MB of RAM. To play Snake Game you will need a minimum CPU equivalent to an Intel Pentium 4 1.40GHz. You will need at least 60 MB of free disk space to install Snake. Snake will run on PC system with Windows 7 and upwards.



Snake Game : minimum requirements

Memory: 256 MB

Graphics Card: NVIDIA GeForce 510

CPU: Intel Pentium 4 1.40GHz

File Size: 60 MB

OS: Windows 7

Figure 5: Hardware requirement

* **Software requirements:**

The basic software requirements is to run this snake game are as follows

* Windows: OS 7
* Mac: OSX version 10 or later
* Linux: Any 64-bit Linux OS

Other than this, you need an Java IDE to run this project, There are numerous Java IDE’s available in world. For example , intellij idea, eclipse IDE, Net Beans, VS Code etc,.

The IDE that ,I have developed and tested this game is Eclipse IDE for java Developers.



Figure 6: Icon of Eclipse IDE Figure 7:Icon of Windows 11

**2.Algorithm and Flowchart:**

1. Select which **Java IDE** that you are going to develop this project.
2. After selecting which **Java IDE** , create new fresh Project and name it as **Snake**.
3. In **Snake project**-**>src** (source) part ,create 3 classes namely **SnakeGame** which containis the **Main**, **GameFrame** and **GamePanel.**
4. After that, create an instance of the **GamePanel** class in **GameFrame** class and similarly create an instance of the **GameFrame** class in **SnakeGame** class inside the “main()”.
5. And then extend the **GameFrame** class by **JFrame** &**Game Panel** class by **JPanel** by importing necessary packages like **java.swing.\*, java.util.Random, java.awt.event.\*** etc.
6. Now, come to **Gameframe** class and start writing the codes for which we want to get displayed on the screen like **setTitle(), setResizable() and setDefaultCloseoperation()** in addition with **this** keyword.
7. Next, come to **GamePanel** class and start breaking down the process into various methods such as **draw(), move(),gameOver(),checkApples(),actionPerformed ()** etc,.
8. After creating that , start to create inner class **MyKeyAdapter** which will extend **KeyAdapter ,** which will have an overridemethod namely **keyPressed()**
9. Now after creating the required methods and inner classes , now start to write the necessary static variables like **screen\_width, screen\_height, unit\_size ,game\_units** and set time delay variable named **Delay**
10. After creating those static variables start writing the integer type variables like ,two separate arrays for the condition to hold the **X** and **Y** coordinates namely **x[] and y[] ,bodypart** (snake’s initial body size) ,**appleseaten** (to count the no. of apples eaten),**appleX** & **appleY** (to make the apple appear randomly)
11. Now create a character variable for to control the **direction** and initially make it as to move in **right side**
12. Next , create a Boolean variable running = false
13. Finally ,declare **timer** and **random** variable for random generation of apples
14. Now create an instance of the random class and setNewDimensions, setBackGround color and other start calling **startGame(),** in **GamePane**l constructor
15. Now , come to **startGame()** , and start declaring necessary things such as a function call for **newApple()** , running=true , timer was supplied with  **delay** value and now end it with **timer.start ()**
16. Now come to paintComponent() and start declaring an method with super keyword paintComponent) and calling the next method draw( e) with a parameter is passed on to it.
17. Now ,in draw() , write an for loop that will create grid lines in our output screen , (this part is completely optional) and use **drawLine()**  (pass drawLine method for both X and Y coordinate) to draw the grid lines over the output screen.
18. Now start to declare the necessary functions to be done in **newApple().** The purpose of this **newApple()** is to create new apples randomly in the screen , once after scoring each points . For that use random variable **appleX (width) and appleY(height)** and cast it as integer type
19. Now come back to **draw()** and start initializing the color for apple as red and color for snake as green using **setColor()**
20. Now come to **move()** ( this method is used to make the snake move & control the movement of the snake) and define a for loop that will make the size of the snake to get increased each and every time the snake eats an apple .For that make shifting of arrays both in x and y direction.
21. Now ,in the same **move()** , create a **switch** condition that will enable the user to control the direction of the movement of the snake (**Left,Right,Down,Up)**
22. Now once again come back to **draw ()** , for to create the body and head of the snake. For that create a for loop , and inside that check whether if it is head or body using **if&else** condition statements . If **(i==0)** means we are dealing with the head part and give green color for it and give rectangle shape , else we are dealing with the body part and give random(R,G,B) values for it .After this also we won’t be able see the movement because we still didn’t define the **move()** properly.
23. Now go back to **actionPerformed()** ,define a if clause with function calling to move(), checkApple() ,checkCollision().
24. After that,now come to **checkCollision()** to define the collisions conditions of the snake . Define a for loop ,which checks if head collides with the body and similarly give 4 if conditions that will check whether the snake hits the left,right,down,up borders
25. Now come to **MyKeyAdapter** inner class to control the movement of the snake .Now inside the **keyPressed()** ,create **switch** condition that will basically works on the **getKeyCode()** , which will get input commands from the user .In that switch statement we should give 4 If condtions work on the given direction inputs. If the direction is not Right , then Left ,If direction is not Left ,then Right .Similarly if direction is not down , then up , and if direction is not Up then Down
26. Next, we should start to work on to make the snake to eat that apple. For that go to **checkApple(),**check if snake’ head and apple meet , the snake’s **body part** should be incremented and also the **appleeaten** count also should be incremented for the case of displaying the **score** and we should call the newapple() that will create new apples after the score gets increased .
27. Copy the checkApple() and Now once again come back to draw method and paste it inside a if clause and write an else clause for it. The Else part is basically means that we had came to the end point and we should start to display the score ,Game over and who presented it.
28. Now go to the **gameOver()** and start writing codes that will get displayed once the user reach the gameOver part .In that give fontSize, color of the font and a font metrics that will make the text should be at the center. And finally declare **drawString()** that will display the GameOver and score in the screen
29. After doing all this steps , you should be able to play the **Snake game** that you have developed.

**FlowChart for the Snake\_Game**

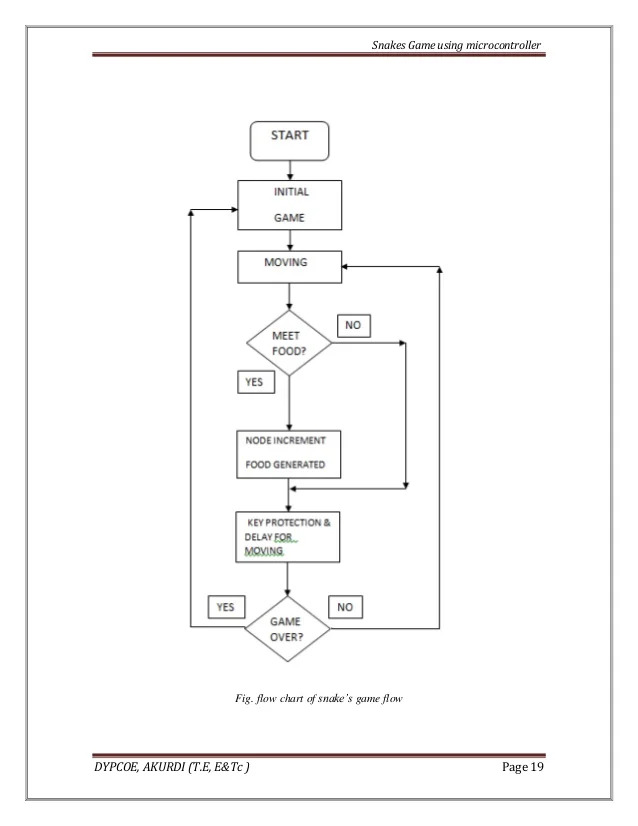
****

Figure 8:Flow Chart of Snake game

**3.Source Code:**

* SnakeGame
* GameFrame
* GamePanel

i.SnakeGame (class that contains main):

public class SnakeGame{

public static void main(String[] args) {

new GameFrame();

}

}

ii.GameFrame:

import javax.swing.JFrame;

public class GameFrame extends JFrame{

GameFrame(){

this.add(new GamePanel());

this.setTitle("Snake");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setResizable(false);

this.pack();

this.setVisible(true);

this.setLocationRelativeTo(null);

}

}

iii.GamePanel:

import javax.swing.\*;

import java.awt.event.\*;

import java.security.Key;

import java.awt.\*;

import java.util.Random;

public class GamePanel extends JPanel implements ActionListener{

static final int SCREEN\_WIDTH=1300;

static final int SCREEN\_HEIGHT=750;

static final int UNIT\_SIZE=50;

static final int GAME\_UNITS=(SCREEN\_WIDTH\*SCREEN\_HEIGHT)/UNIT\_SIZE;

static final int DELAY=100;

final int x[]=new int[UNIT\_SIZE];

final int y[]=new int[UNIT\_SIZE];

int bodyParts=6;

int applesEaten;

int appleX;

int appleY;

char direction='R';

boolean running=false;

Timer timer;

Random random;

GamePanel(){

random=new Random();

this.setPreferredSize(new Dimension(SCREEN\_WIDTH,SCREEN\_HEIGHT));

this.setBackground(Color.black);

this.setFocusable(true);

this.addKeyListener(new MyKeyAdapter());

startGame();

}

public void startGame() {

newApple();

running=true;

timer=new Timer(DELAY,this);

timer.start();

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

draw(g);

}

public void draw(Graphics g) {

if(running) {

/\*for(int i=0;i<SCREEN\_HEIGHT/UNIT\_SIZE;i++) {

g.drawLine(i\*UNIT\_SIZE, 0, i\*UNIT\_SIZE, SCREEN\_HEIGHT);

g.drawLine(0, i\*UNIT\_SIZE, SCREEN\_HEIGHT, i\*UNIT\_SIZE);

}\*/

g.setColor(Color.red);

g.fillOval(appleX, appleY, UNIT\_SIZE, UNIT\_SIZE);

//creating the body of the snake

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

//if: for if it is the head of the snake

if(i==0) {

g.setColor(Color.green);

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

}

//else: for if it is the body part of the snake

else {

//Creating a random RGB color type

g.setColor(new Color(45,180,0));

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

}

}

g.setColor(Color.red);

g.setFont(new Font("Bold",Font.BOLD,40));

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

g.drawString("SCORE:"+applesEaten,(SCREEN\_WIDTH-metrics.stringWidth("SCORE:"+applesEaten))/2,g.getFont().getSize());

}

else {

gameOver(g);

}

}

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 checkApple() {

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

bodyParts++;

applesEaten++;

newApple();

}

}

public void checkCollisions() {

for(int i=bodyParts;i>0;i--) {

//if:snake's head and body collides

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

running=false;

}

}

//checks if head touches the left border

if(x[0]<0) {

running=false;

}

//checks if head touches the right border

if(x[0]>SCREEN\_WIDTH){

running=false;

}

//checks if head touches the top border

if(y[0]<0) {

running=false;

}

//checks if head touches the bottom border

if(y[0]>SCREEN\_HEIGHT){

running=false;

}

if(!running) {

timer.stop();

}

}

public void gameOver(Graphics g) {

//Score Display

g.setColor(Color.white);

g.setFont(new Font("Bold",Font.BOLD,40));

FontMetrics metrics1=g.getFontMetrics(g.getFont());

g.drawString("SCORE:"+applesEaten,(SCREEN\_WIDTH-metrics1.stringWidth("SCORE:"+applesEaten))/2,g.getFont().getSize());

//for Game over indication

g.setColor(Color.red);

g.setFont(new Font("Ink Free",Font.BOLD,75));

FontMetrics metrics2=g.getFontMetrics(g.getFont());

g.drawString("GameOver",(SCREEN\_WIDTH-metrics2.stringWidth("Game Over"))/2, SCREEN\_HEIGHT/2);

g.setColor(Color.WHITE);

g.setFont(new Font("Italic",Font.BOLD,50));

FontMetrics metrics3=g.getFontMetrics(g.getFont());

g.drawString("Presentedby:2021PITCS208",(SCREEN\_WIDTH- metrics3.stringWidth("Presented by: 2021PITCS208"))/2,500);

}

@Override

public void actionPerformed(ActionEvent e) {

if(running){

move();

checkApple();

checkCollisions();

}

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;

}

}

}

}

**4.Actual Output with necessary Screenshots:**

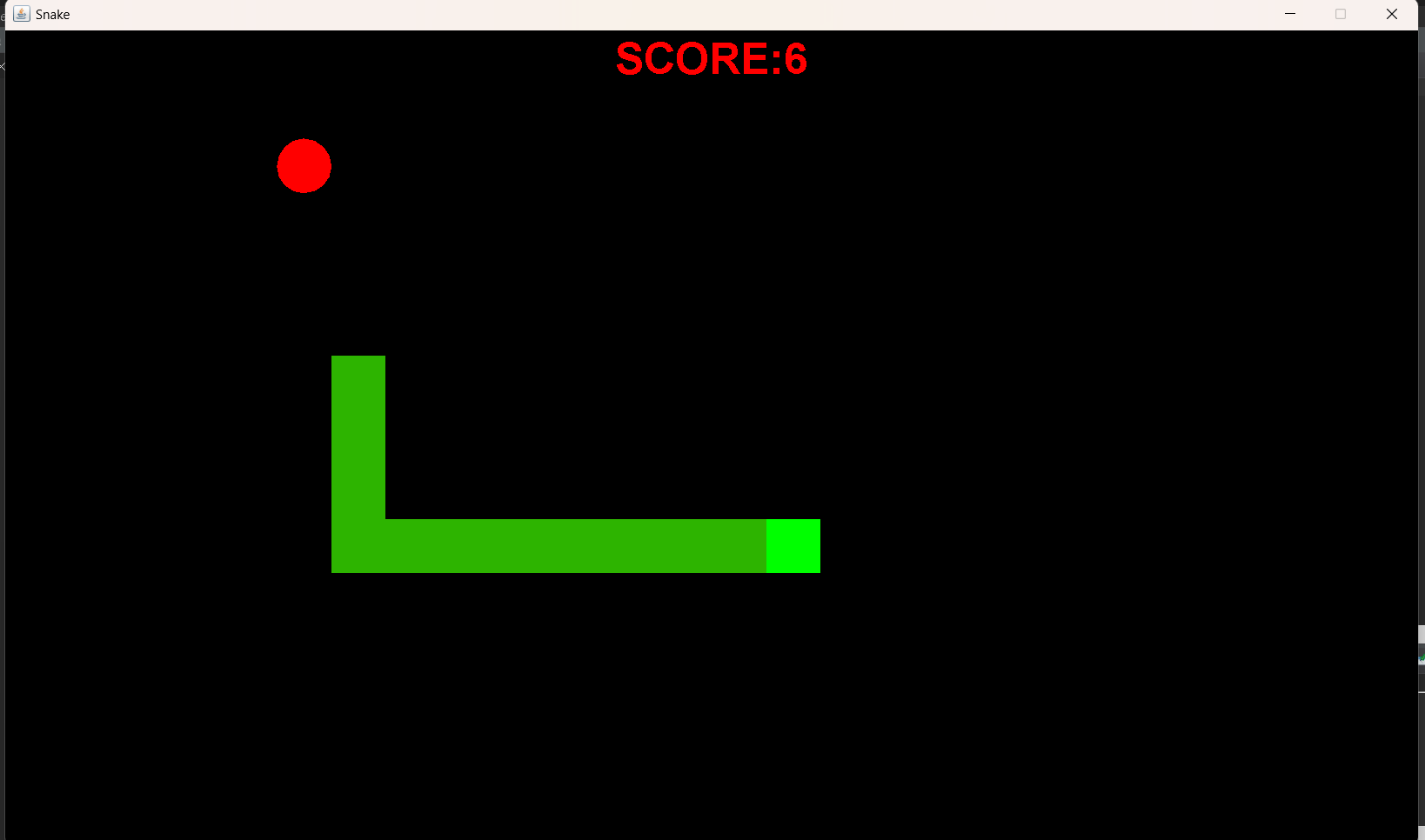


Figure 9: Sample 1

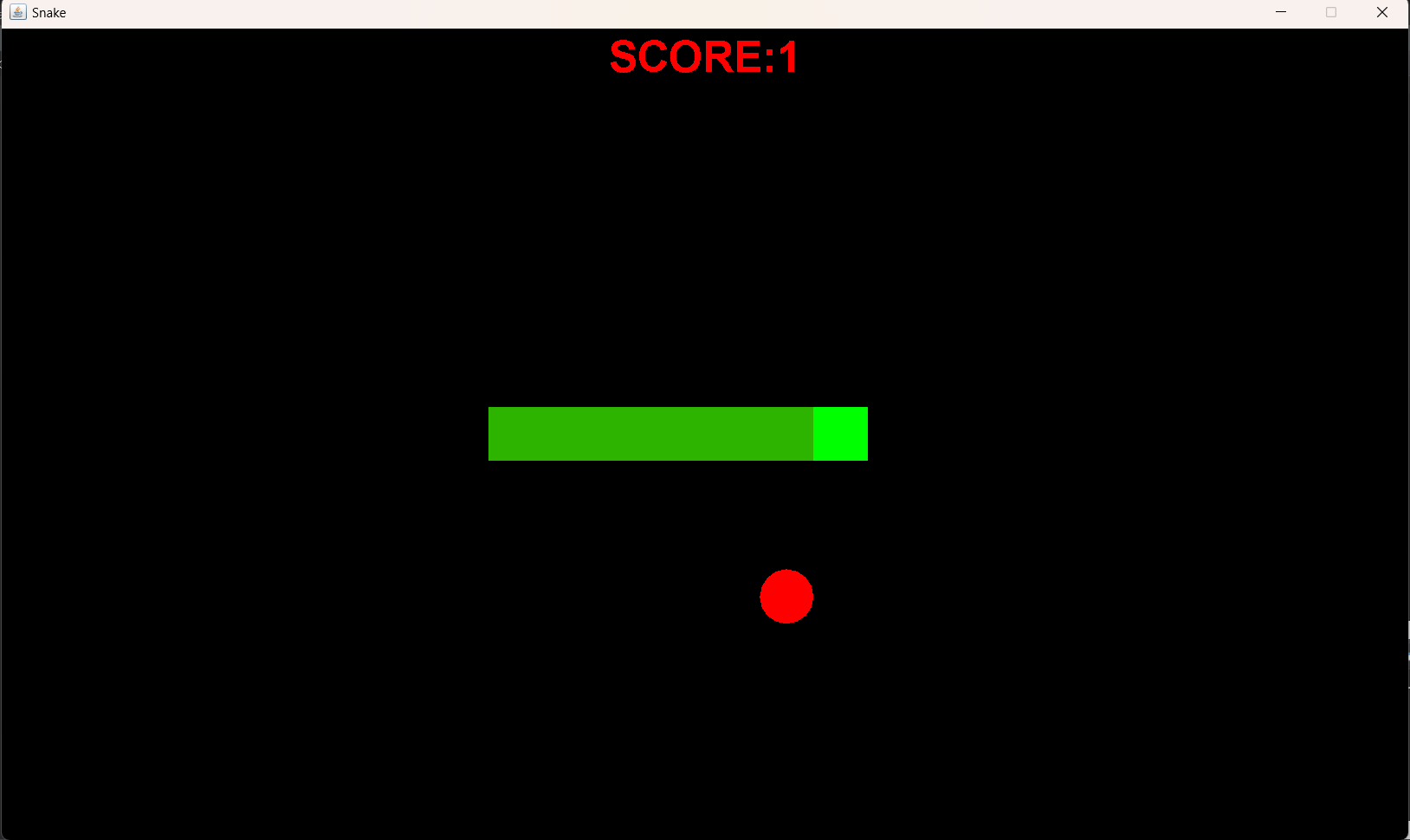


Figure 10: Sample 2

Figure 11: Sample 3



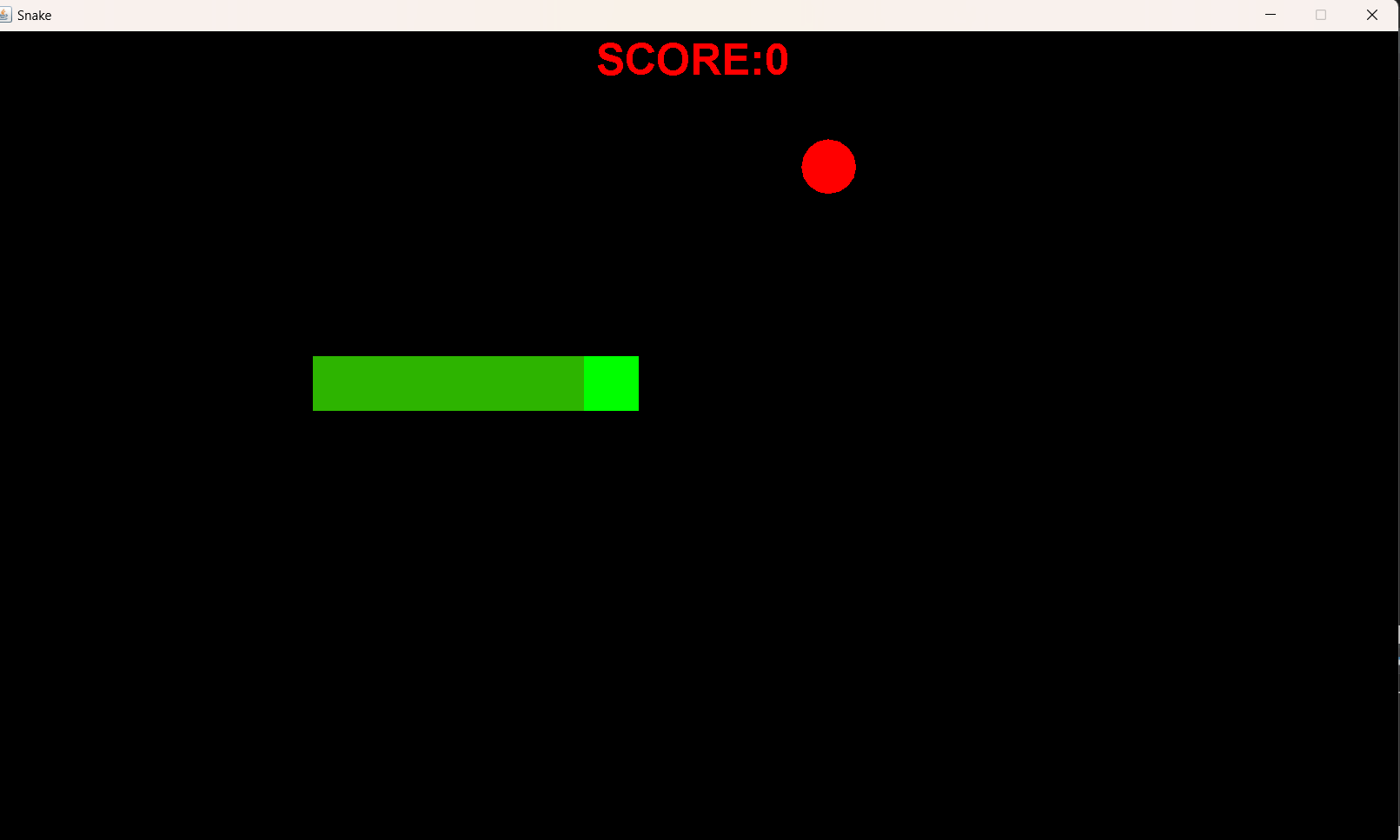


Figure 12: Sample 4

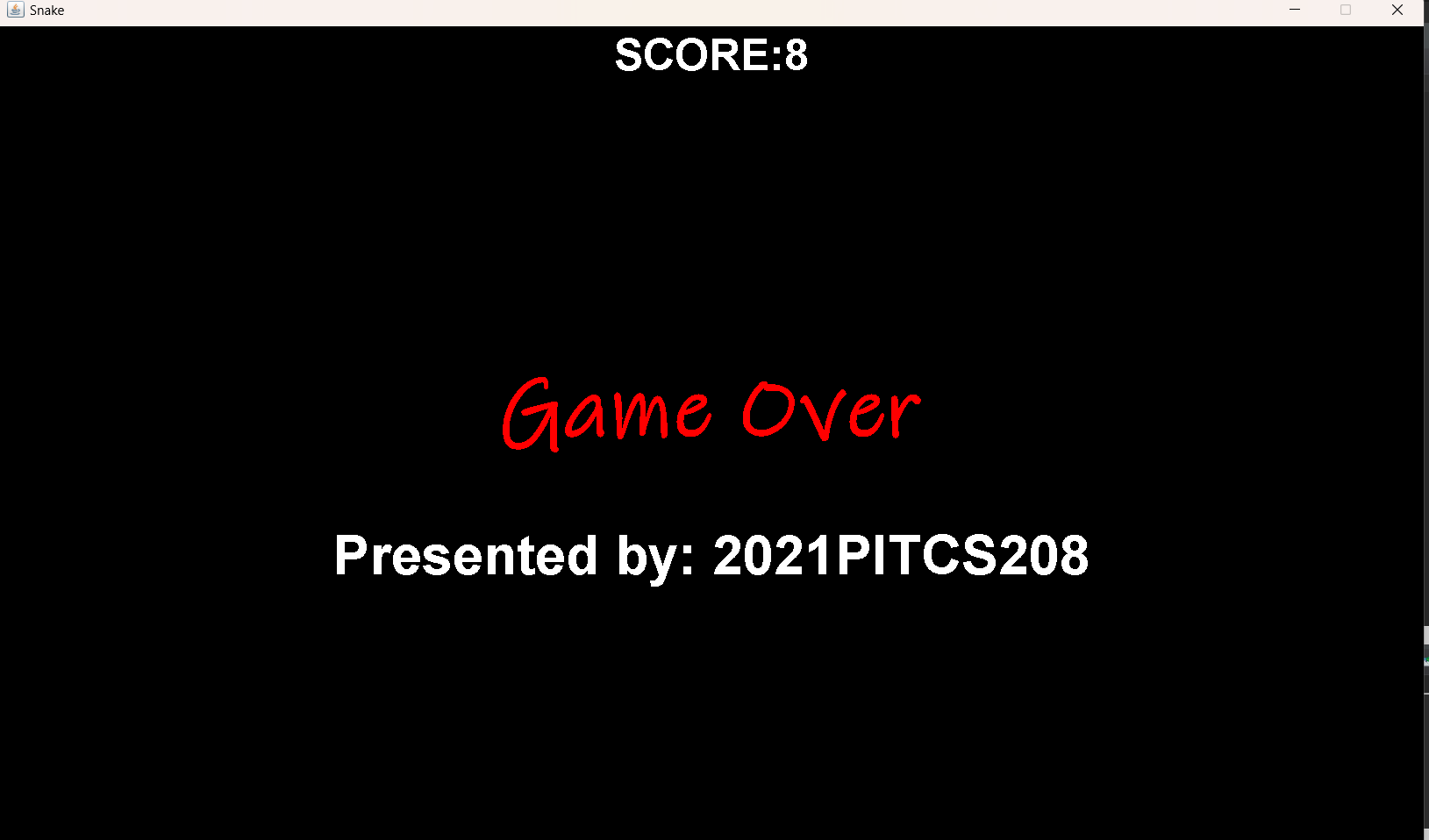


Figure 13: Sample 5

**Result:**

Thus a Mini Project of a Snake Game was created using Java IDE in addition with GUI was developed & executed and even the output is performed and documentation is done successfully.