**SNAKE GAME**

**A** **Project** **Work** **Report**

*Submitted* *in* *the* *partial* *fulfilment* *for* *the* *award* *of* *the* *degree* *of*

**BACHELOR** **OF** E**NGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION IN BIG DATA ANALYTICS**

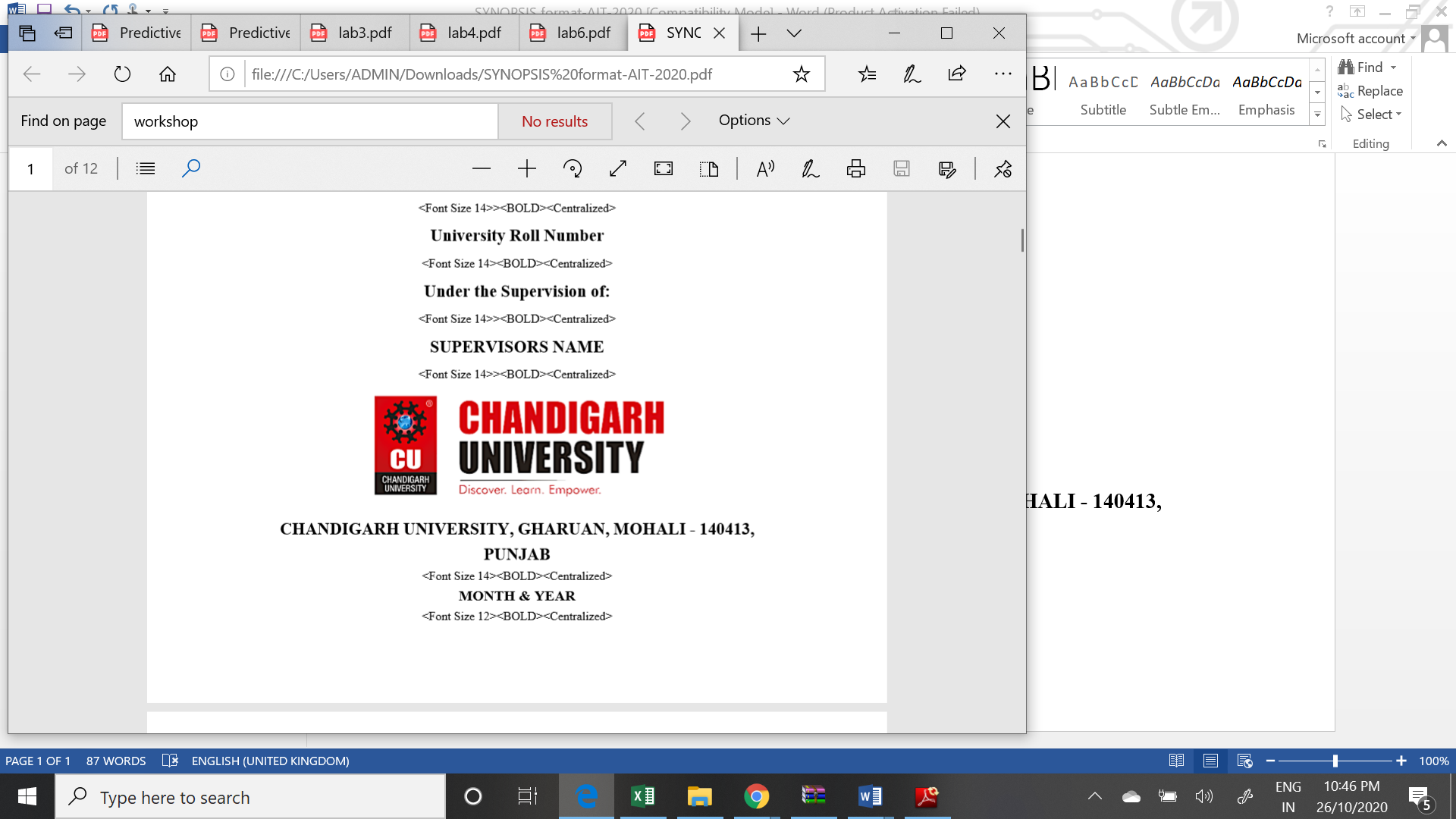
**Submitted** **by:**

**PRABHAT RAI**

**19BCS3761**

**Under** **the** **Supervision** **of:**

**Ms. Jyoti Mehra**



**CHANDIGARH** **UNIVERSITY,** **GHARUAN,** **MOHALI** **-** **140413,** **PUNJAB**

**December, 2020**

**Name and signature of student(s)**

Prabhat – 19BCS3761

Harshvardhan – 19BCS3752

Bhumika – 19BCS3776

Dhananjay – 19BCS3806

**Name and signature of Supervisor**

Ms. Jyoti Mehra

# 

# PROJECT COMPLETION CERTIFICATE

## Project Title

This is to certify that the Prabhat Rai has successfully completed the project work titled “Snake Game” Submitted in the partial fulfilment for the award of the degree of**BACHELOR** **OF** E**NGINEERING** **IN COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION IN BIG DATA ANALYTICS.**

This project is the record of authentic work carried out during the academic year

**2020-2021**

Ms. Jyoti Mehra

**Date: 08-12-2020**

# DECLARATION

I the undersigned solemnly declare that the project report is based on my own work carried out during the course of our study under the supervision of Ms. Jyoti Mehra. I assert the statements made and conclusions drawn are an outcome of my work. I further certify that the work contained in the report is original and has been done by me under the general supervision of my supervisor.

II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.

III. We have followed the guidelines provided by the university in writing the report.

IV. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

**Prabhat Rai**

**19BCS3761**

# ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to Ms. Jyoti Mehra for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents and my department for their kind co-operation and encouragement which help me in completion of this project.

THANKS AGAIN TO ALL WHO HELPED

**INDEX**

Chapter 1: Introduction

Chapter 2: Project Requirements (Software/Hardware requirements)

Chapter 3: Implementation (Algorithm, code)

Chapter 4: Output Analysis (screenshots)

Chapter 5: References

**Introduction**

We have created a project on “Snake Game” using Java programming. As in our childhood we have played this game a lot. We used to enjoy it very much. So, we want you to play this game and revive your childhood. It was generally being played on game machines as an arcade. After that the mobile phone manufacturing company ‘Nokia’ started to include this game in their phones. This game has improved in these years but the enthusiasm of playing this game has been declined. As now there are huge game with excellent graphics and performance and people are playing those. But there are some people who like to play arcade games like snake, tic-tac-toe, etc. And our motive is to recreate your childhood memories. So, play this game and enjoy the implemented graphical interactivity provided by us.

**Project Requirements**

**System Requirement:**

**Hardware:**

* Intel Pentium IV or above/ Ryzen 3 or above
* Memory 1GB minimum (1.25GB or more Recommended)
* Disk space 500MB minimum (1GB or more Recommended)
* Processor speed 1GHz minimum (1.5GHz or more Recommended)

**Software:**

* **Operating System:**

Windows (7, 8, 8.1 or 10)

Linux (Any)

Mac OS X or above

* **JDK:**

JDK (8 or above)

* **JRE:**

JRE (1.8.0\_241 or above)

**Development Tools:**

* + - * Java Swing (Eclipse or Visual Studio Code)
      * Java AWT (Eclipse or Visual Studio Code)

**Implementation**

**Algorithm:**

**Step 1:** Creating a Frame/Active window.

* The frame supports the Panel on which the game will run.
* Can be created by inheriting the “JFrame” class.
* JFrame can be found in SWING as well as AWT.
* The following code snippet will create the Frame.

import javax.swing.JFrame;

OR

import java.awt.Frame;

Class GameFrame extends JFrame{

}

**Step 2:** Creating the Panel on which the game will run.

* The Panel will support certain objects like graphics,time,visuals etc.
* The Panel and its functionality can be used by inheritng the JPanel class.
* The JPanel can be also be imported from SWING as well as AWT.
* The following code snippet creates the panel.

Import javax.swing.JPanel;

OR

Import java.awt.event.\*;

Import java.awt.\*;

**Step 3:** Setting the panel

* + Specify dimensions, background colour,any text to be displayed on panel.
  + Setting timer object .
  + Adding input methods.
  + Following is the required code:-

GamePanel(){

Random = new Random();

This.setPreferredSize(newDimension(SCREEN\_WIDTH, SCREEN\_HEIGHT));

This.setBackground(Color.black);

This.setFocusable(true);

This.addKeyListener(new MyKeyAdapter());

startGame();

}

**Step** **4:** Defining a way to bring the our active code into actionon the Panel.

* + For this we will implement the interface ActionListener in our Panel.
  + The ActionListener function- actionPerformed(ActioEvent) will provide the changing fuctionality to our panel.
  + The function actionPerformed must be implemented else error is shown.
  + The following code snippet will perform the above:-

Public class GamePanel extends JPanel implements ActionListener{

@Override

Public void actionPerformed(ActionEvent arg0) {

//Code required

}

}

**Step 5:** Providing input mechanism.

* We are providing input through keyboard.
* Keyboard inputs can be extending the - “ keyAdapter” class.
* Then we will override the keyPressed(keyEvent) function of this class and define the keys for specific tasks.
* The above functionality can be implemented by following code snippet:

Public class MyKeyAdapter extends KeyAdapter{

@Override

Public void keyPressed(KeyEvent e) {

Switch(e.getKeyCode()) {

//required fields inside

}

}

**Code:**

**import** java.awt.Color;

**import** java.awt.Dimension;

**import** java.awt.Font;

**import** java.awt.FontMetrics;

**import** java.awt.Graphics;

**import** java.awt.Image;

**import** java.awt.Toolkit;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.awt.event.KeyAdapter;

**import** java.awt.event.KeyEvent;

**import** java.net.URL;

**import** java.util.Random;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.Timer;

**public** **class** GamePanel **extends** JPanel **implements** ActionListener{

**static** **final** **int** ***SCREEN\_WIDTH*** = 600;

**static** **final** **int** ***SCREEN\_HEIGHT*** = 600;

**static** **final** **int** ***UNIT\_SIZE*** =25;

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

**static** **int** *DELAY* = 75;

**final** **int** x[] = **new** **int**[***GAME\_UNITS***];

**final** **int** y[] = **new** **int**[***GAME\_UNITS***];

**int** bodyParts = 4;

**int** applesEaten;

**int** appleX;

**int** appleY;

**int** Score = applesEaten;

**int** HighScore = -1;

**char** direction = 'R';

**boolean** running = **false**;

Timer timer;

Random random;

**static** **boolean** *gameOn* = **false**;

**private** Image gameImage = **null**;

**private** **boolean** isStartImage = **true**;

Graphics g;

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.restart();

}

**public** **void** pause() {

GamePanel.*gameOn* = **true**;

timer.stop();

}

**public** **void** reset() {

GameFrame n=**new** GameFrame();

n.add(**new** GamePanel());

n.setTitle("Snake");

n.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

n.setResizable(**false**);

n.pack();

n.setVisible(**true**);

n.setLocationRelativeTo(**null**);

bodyParts=4;

Score=0;

startGame();

}

**public** **void** resume() {

GamePanel.*gameOn* = **false**;

timer.start();

}

**public** **void** paintComponent(Graphics g) {

**super**.paintComponent(g);

draw(g);

}

**public** **void** draw(Graphics g) {

**if**(running) {

//can be used to create matrix to check size of snake and fruit

/\*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\_WIDTH, i\*UNIT\_SIZE);

}\*/

**if**(isStartImage)

{

drawImage(g);

}

**else** {

g.setColor(Color.***red***);

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

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

**if**(i == 0) {

g.setColor(Color.***green***);

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

}

**else** {

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

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

}

}

g.setColor(Color.***red***);

g.setFont(**new** Font("Ink Free", Font.***BOLD***,40));

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

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

}

}

**else** {

gameOver(g);

}

}

**public** **void** drawImage(Graphics g) {

**if**(gameImage == **null**) {

**try** {

URL imagePath = GamePanel.**class**.getResource("Game.png");

gameImage = Toolkit.*getDefaultToolkit*().getImage(imagePath);

}

**catch**(Exception e) {

e.printStackTrace();

}

}

g.drawImage(gameImage, 0, 0, ***SCREEN\_HEIGHT***, ***SCREEN\_WIDTH***, **this**);

}

**public** **void** newApple()//printing apples at random position on the frame

{

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() //checking if apple is eaten

{

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

bodyParts++;

Score++;

*DELAY* = *DELAY* - 2;

newApple();

}

}

**public** **void** checkCollisions() //checking snakes collision

{

//checking if head of snake collides with its body

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

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

running = **false**;

}

}

//checking if head of snake collides with left wall

**if**(x[0] < 0) {

running = **false**;

}

//checking if head of snake collides with right wall

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

running = **false**;

}

//checking if head of snake collides with top wall

**if**(y[0] < 0) {

running = **false**;

}

//checking if head of snake collides with bottom wall

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

running = **false**;

}

**if**(!running) {

timer.stop();

}

}

**public** **void** gameOver(Graphics g) {

//Score

g.setColor(Color.***red***);

g.setFont(**new** Font("Ink Free", Font.***BOLD***,40));

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

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

//Game Over

g.setColor(Color.***red***);

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

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

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

//Reset

g.setColor(Color.***red***);

g.setFont(**new** Font("Ink Free", Font.***BOLD***,40));

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

g.drawString("Press R to Restart", (***SCREEN\_WIDTH*** - metrics3.stringWidth("Press R to Restart"))/2, 3\*(***SCREEN\_HEIGHT***)/4);

//Close the Game

g.setColor(Color.***red***);

g.setFont(**new** Font("Ink Free", Font.***BOLD***,40));

FontMetrics metrics4 = getFontMetrics(g.getFont());

g.drawString("Press C to close the game", (***SCREEN\_WIDTH*** - metrics4.stringWidth("Press C to close the game"))/2, 7\*(***SCREEN\_HEIGHT***)/8);

}

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

**if**(isStartImage==**false**) {

**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**;

}

**case** KeyEvent.***VK\_SPACE***:{

**if**(GamePanel.*gameOn*) {

resume();

} **else** {

pause();

}

**break**;

}

**case** KeyEvent.***VK\_R***:{

reset();

**break**;

}

**case** KeyEvent.***VK\_C***:{

System.*exit*(0);

**break**;

}

**case** KeyEvent.***VK\_ENTER***:{

isStartImage = **false**;

**break**;

}

}

}

}

}

**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**);

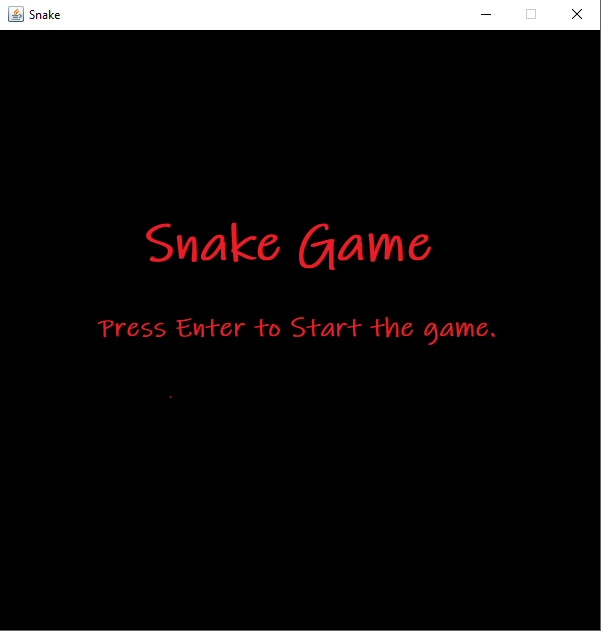
**this**.setLocationRelativeTo(**null**);

}

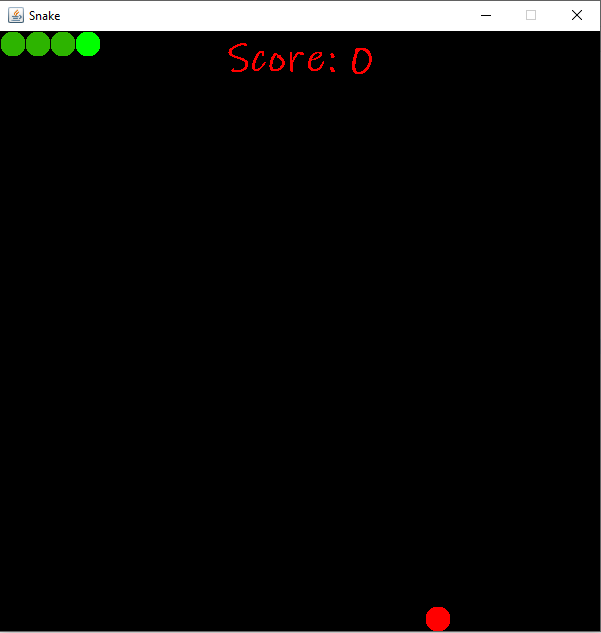
}

**Output Analysis**

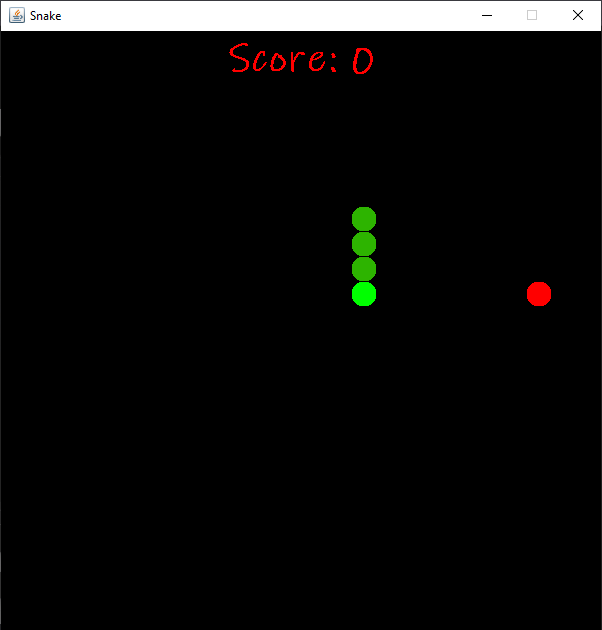
**After execution we have to press ‘Enter’.**

****

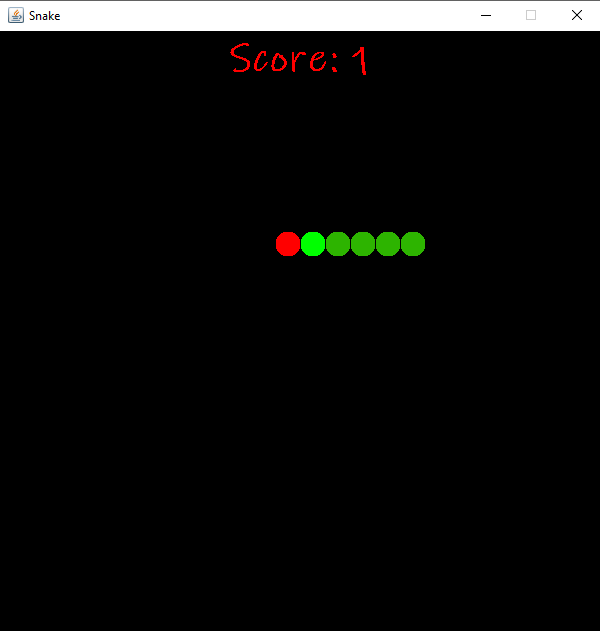
**After pressing ‘Enter’**

****

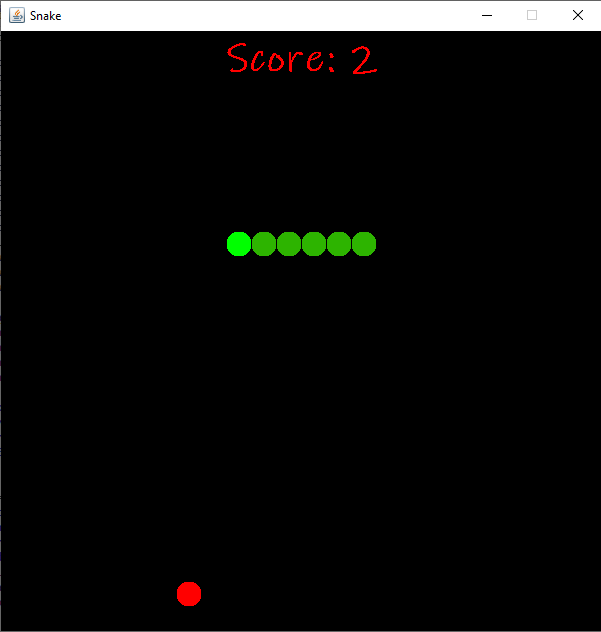
**We use navigation keys to control its movement.**

****

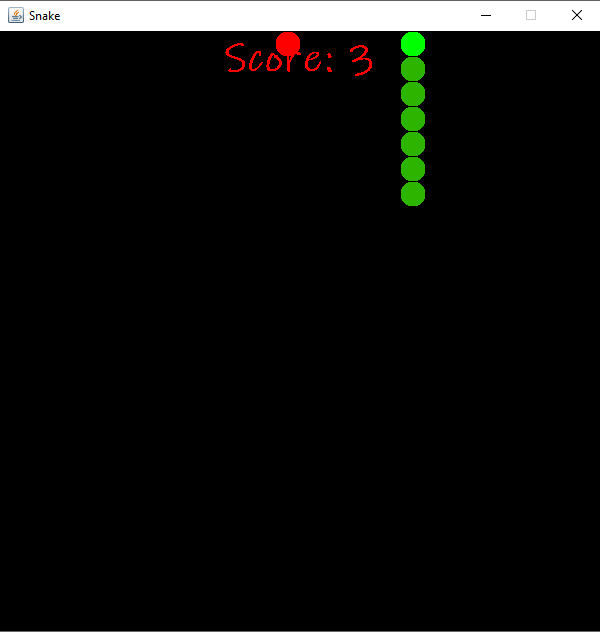
**Snake before eating the food.**

****

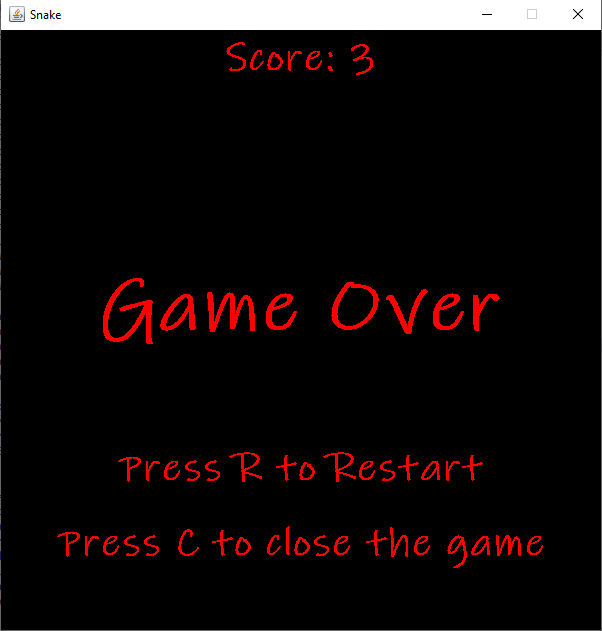
**Sake after eating the food.**

****

**Snake before colliding with wall.**

****

**Snake after colliding with the wall.**

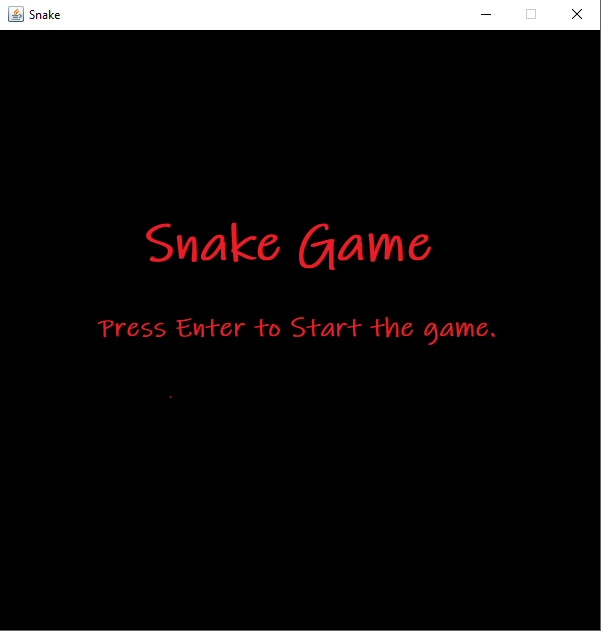
****

**Game Over:** This will happen when snake either collides with walls or its head collides with its other body parts.

**Press R:** This will restart the game.

**Press C:** This will close the panel and the frame ad game will be closed.

**After pressing ‘R’ for restart.**

****

**The game starts again.**

**References**

* https://www.google.co.in
* [https://www.oracle.com/](https://www.oracle.com/%20)
* <https://www.javatpoint.com/java-swing>
* <https://www.javatpoint.com/java-awt>
* <https://eclipse.org/>
* https://www.youtube.com/watch?v=bI6e6qjJ8JQ