A Mini Project

Report On

SNAKE GAME

Submitted in partial fulfillment of requirements for the Course CSE18R272 - JAVA PROGRAMMING

Bachelor's of Technology

In

Computer Science and Engineering

Submitted By

LAKSHMI PRIYANKA
(9920004507)
MEGHANA REDDY
(9920004561)

Under the guidance of

Dr. R. RAMALAKSHMI

(Associate Professor)

Department of Computer Science and Engineering Kalasalingam Academy of Research and Education Anand Nagar, Krishnankoil-626126

DECEMBER 2021

ABSTRACT

SNAKE GAME

This project aims to bring the fun and simplicity of snake game with some new features. It will include computer controlled intelligent opponents whose aim will be to challenge the human players. It will also have the multiplayer feature that will allow more than one players to play the game over a network.

This project explores a new dimension in the traditional snake game to make it more interesting and challenging. The simplicity of this game makes it an ideal candidate for a minor project as we can focus on advanced topics like multiplayer functionality and implementation of computer controlled intelligent opponents.

DECLARATION

I hereby declare that the work presented in this report entitled "**SNAKE GAME**", in partial fulfilment of the requirements for the course CSE18R272- Java Programming and submitted in **Department of Computer Science and Engineering, Kalasalingam Academy of Research and Educa- tion** (**Deemed to be University**) is an authentic record of our own work carried out during the period from **DEC 2021** under the guidance of Mr. **Dr. R. Ramalakshmi** (Associate Professor).

The work reported in this has not been submitted by me for the award of any other degree of this or any other institute.

LAKSHMI PRIYANKA
9920004507
MEGHANA REDDY
9920004561

ACKNOWLEDGEMENT

I express my deep gratitude to Mr.Dr.R.Ramalakshmi for their valuable guidance throughout my training.

LAKSHMI PRIYANKA
9920004507
MEGHANA REDDY
9920004561

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

The history of the Snake game goes back to the 1970's. However, it was the 1980's when 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 game involves controlling a single block or snakehead by turning only left or right by ninety degrees until you manage to eat an food. When you get the food, the Snake grows an extra block or body segment.

If, or rather when, the snake bumps into the edge of the screen or accidentally eats himself the game is over. The more apples the snake eats the higher the score.

Objectives

List the objectives of the project work...

1. To develop a code..
2. To implement a project for ...

Chapter 2

PROJECT DESCRIPTION

This **Mini Project in java 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 eat. Every time the snake eats the food, its length will by increased by one element along with the score.

System Architecture:

System architecture is the skeleton view behind the GUI part of a game. System architecture defines the working methodology of the game and shows the components, their relationships and how they evolve to make the game work. The system architecture of this particular game can be divided into two parts:

- Game components.
- Game architecture.

Game Component:

Game Components are behind the scene classes and methods which make up the game and all the functionalities. The components are mesh together and there are lot of inter relationships between them. The main components are described below:

Main class: Main class is where all the components are connected and it's the heart and mind of the game. In this game it is named " mMain.cs^ prime . The GUI of the game is represented in this class. The movement is also listened from here & different methods are used for different functionality.

Snake class: Snake class is a class which is used to draw, design, control movement and define Snake behavior.

Snakel class: Snakel class is a class which is used to draw, design, control movement and define Snakel behavior.

Here some method of Snake & Snakel: Public void Draw ()-for draw Snake
Public void Draw (Graphics graphics)-for snake color Public void Move (int direction)-for move
Public void Grow ()-for snake grow

Food class: Food class is a class which is used to draw, design food. Here food is randomly generated. When the Snake & Snakel eat food

they become grow.

Public Food (Random rand)

Public void Draw (Graphics graphics) Public void Generate (Random rand)

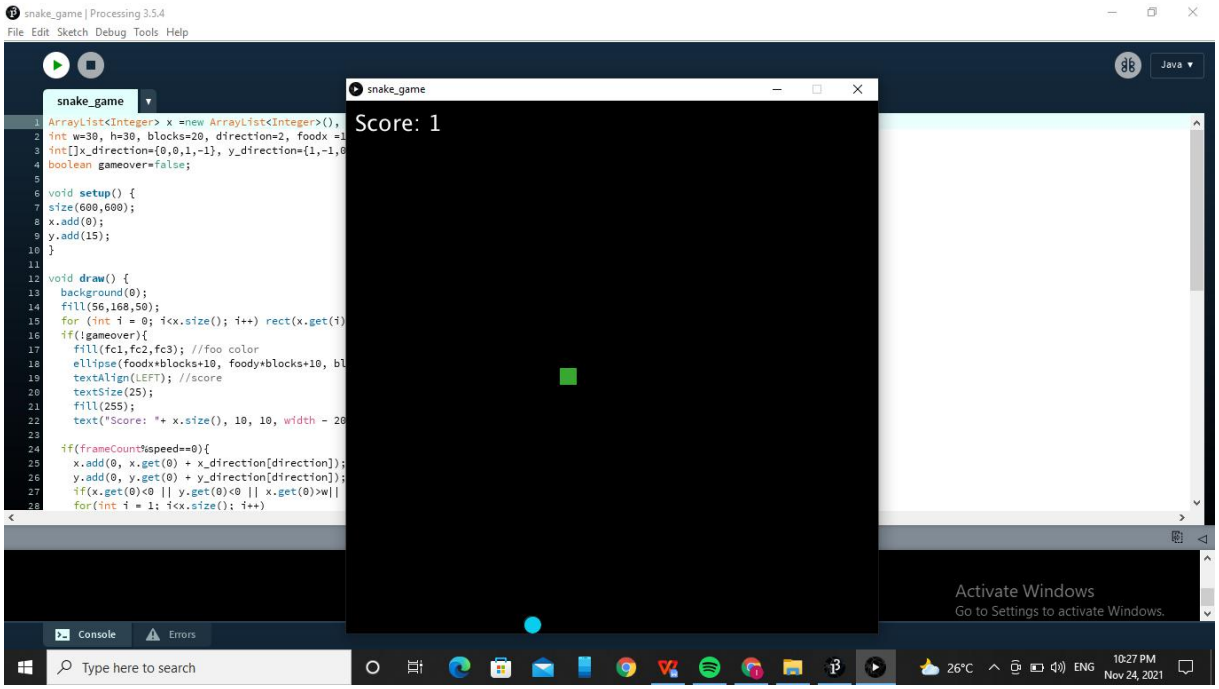
x = Next(0,30)^ * 10 ;

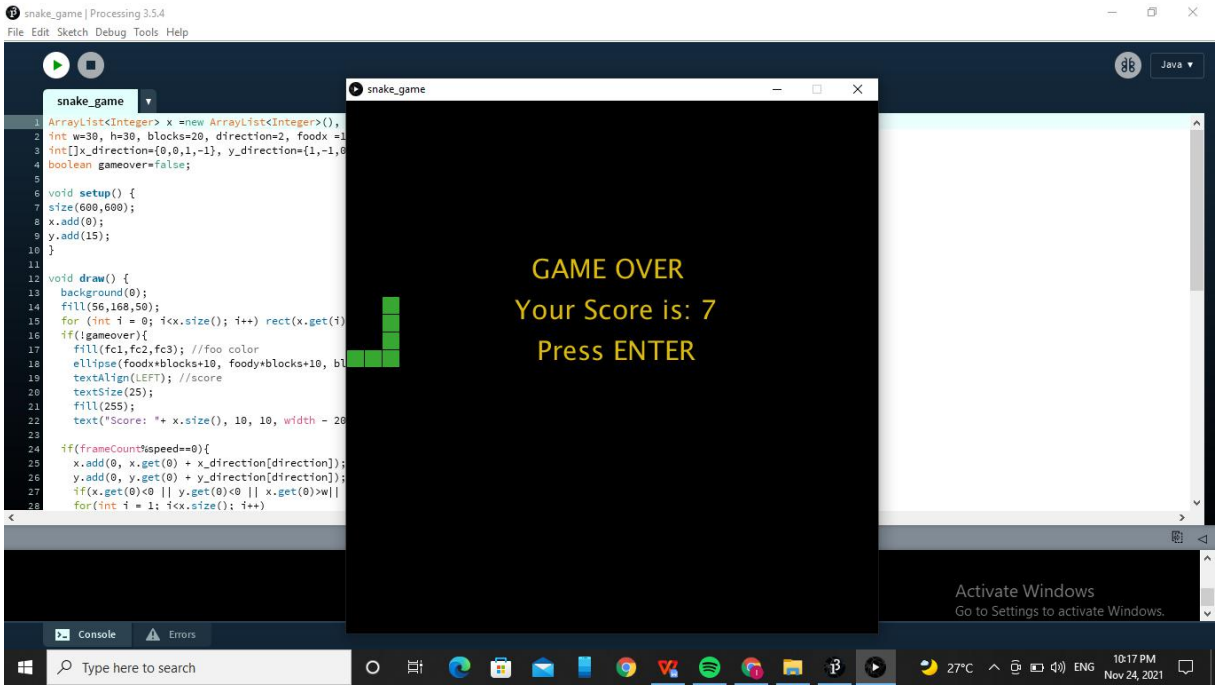
y = nd.Next(0,20)*10 ;

Game Architecture: The Game Architecture is the simplified graphical view of the game. It shows how the components work and the basic view of the game at action. The architectural view of the game is very important. Simply it gives an overview of the game functionality and it makes the game easy to understand.

Chapter 3

OUTPUT SCREENSHOT





Chapter 4

CONCLUSION

Here, I conclude my lines of my term paper on the topic ‘snake game’ with the extreme satisfaction and contentment. This term paper contains brief definition of snake game together with its features and functions. Added to this, my term paper contains the basic description to Create, Edit, Save, Delete and Exit from file through C program. It also includes practical implementation of text editors through complex c program which is created by our group.

SOURCE CODE

```
ArrayList<Integer> x =new ArrayList<Integer>(), y = new
ArrayList<Integer>();
int w=30, h=30, blocks=20, direction=2, foodx =15, foody = 15, fc1 = 255,
fc2 = 255, fc3 = 255, speed = 8;
int[]x_direction={0,0,1,-1}, y_direction={1,-1,0,0};
boolean gameover=false;

void setup() {
size(600,600);
x.add(0);
y.add(15);
}

void draw() {
  background(0);
  fill(56,168,50);
  for (int i = 0; i<x.size(); i++) rect(x.get(i)*blocks,y.get(i)*blocks, blocks,
blocks);
  if(!gameover){
    fill(fc1,fc2,fc3); //foo color
    ellipse(foodx*blocks+10, foody*blocks+10, blocks, blocks); //food
    textAlign(LEFT); //score
    textSize(25);
    fill(255);
    text("Score: "+ x.size(), 10, 10, width - 20, 50);

  if(frameCount%speed==0){
    x.add(0, x.get(0) + x_direction[direction]);
    y.add(0, y.get(0) + y_direction[direction]);
    if(x.get(0)<0 || y.get(0)<0 || x.get(0)>w|| y.get(0)>h) gameover=true;
    for(int i = 1; i<x.size(); i++)
    if(x.get(0)==x.get(i)&&y.get(0)==y.get(i)) gameover = true;
    if(x.get(0)==foodx && y.get(0)==foody){
      if(x.size()%5==0 && speed>2) speed-=1; //speed increase
      foodx = (int)random(0,w);
      foody = (int)random(0,h);
      fc1 = (int)random(255); fc2 = (int)random(255); fc3 =
(int)random(255);
```

```
    }else{
    x.remove(x.size()-1);
    y.remove(y.size()-1);
  }
  }
  }else{
    fill(219, 186, 18);
    textSize(30);
    textAlign(CENTER);
    text("GAME OVER \n Your Score is: "+x.size() +"\n Press ENTER",
width/2, height/3);
    if(keyCode == ENTER){
      x.clear();
      y.clear();
      x.add(0);
      y.add(15);
      direction = 2;
      speed = 8;
      gameover = false;

    }
  }}

void keyPressed(){
  int newdirection=keyCode == DOWN?  0:(keyCode== UP?1:(keyCode
== RIGHT?2:(keyCode == LEFT?3:-1)));
  if (newdirection !=-1) direction = newdirection;
```