PROJECT REPORT

Submitted by

MOHAMED HARSHAD M - RA2211003010152
DHANUSH A - RA2211003010158

Under the Guidance of

DR. SARAVANAN S
Assistant Professor
COMPUTING TECHNOLOGY

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING
(CORE)

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

SRM INSTITUTION OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this Project Report titled **"C++ Snake Game"** is the bonafide work done by MOHAMED HARSHAD M - RA2211003010152 and DHANUSH A - RA2211003010158 who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

DR. SARAVANAN S

OODP – Course

Faculty Assistant

Professor

Department of

Computing Technology

SRMIST

SIGNATURE

DR. PUSHPALATHA M

Head of the Department

Department of Computing

Technology

SRMIST

SIGNATURE

TABLE OF CONTENTS

PROBLEM STATEMENT:

Making a C++ snake game using Object Oriented Programming Concepts and Design Concepts

CONCEPTS USED:

To make a snake game using C++ OODP (Object Oriented Design Principles), we use the following concepts:

1.Classes: We use classes to define the basic components of the game, such as the Snake, Game, and Food classes.

2.Encapsulation: We use encapsulation to hide the implementation details of each class and provide a

public interface for accessing its methods and attributes.

3. Inheritance: We can use inheritance to create subclasses that inherit the attributes and methods of their parent classes. For example, we can create a subclass of the Snake class that has additional functionality, such as the ability to move faster.

4.Abstraction: We use abstraction to simplify the design of the game by representing complex systems with simpler models. For example, we can represent the snake as a series of connected line segments rather than a complex set of movements and collisions.

5.Modularity: We use modularity to break down the game into smaller, more manageable components that can be developed and tested independently. For example, we can develop the Snake class and the Game class separately and then integrate them into the final game.

MODULES OF PROJECT:

In the constructor of the Snake class, we initialize the length to 1, the direction to 0 (up), and add the initial position of the snake to the body vector.

The **move()** method updates the position of the snake by moving each segment of the snake's body forward one space, starting with the last segment and ending with the head. Then, it moves the head of the snake in the current direction based on the value of the direction attribute. The directions are defined as follows: 0 for up, 1 for right, 2 for down, and 3 for left.

The **grow()** method adds a new segment to the snake's body by duplicating the position of the last segment and adding it to the end of the body vector. It also increments the length attribute.

The **checkCollision()** method checks whether a given position collides with the snake's body. It does this by

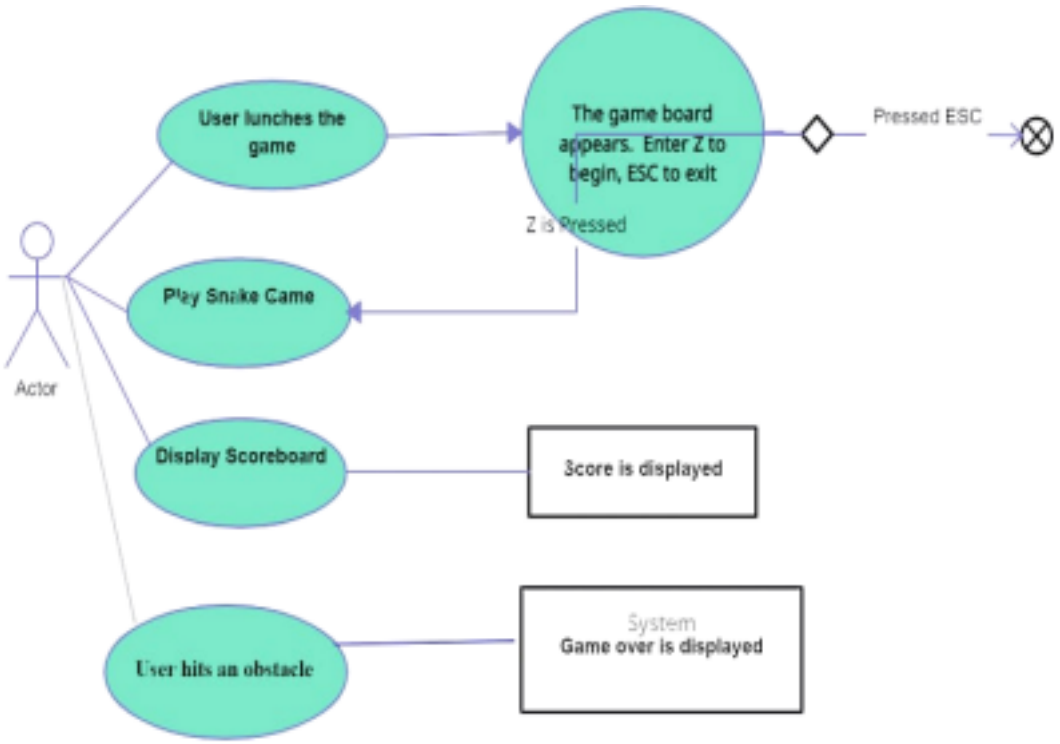
iterating over each segment of the snake's body and checking whether its position matches the given position.

The **setDirection()** method updates the direction attribute to the given value if it is a valid direction (0-3).

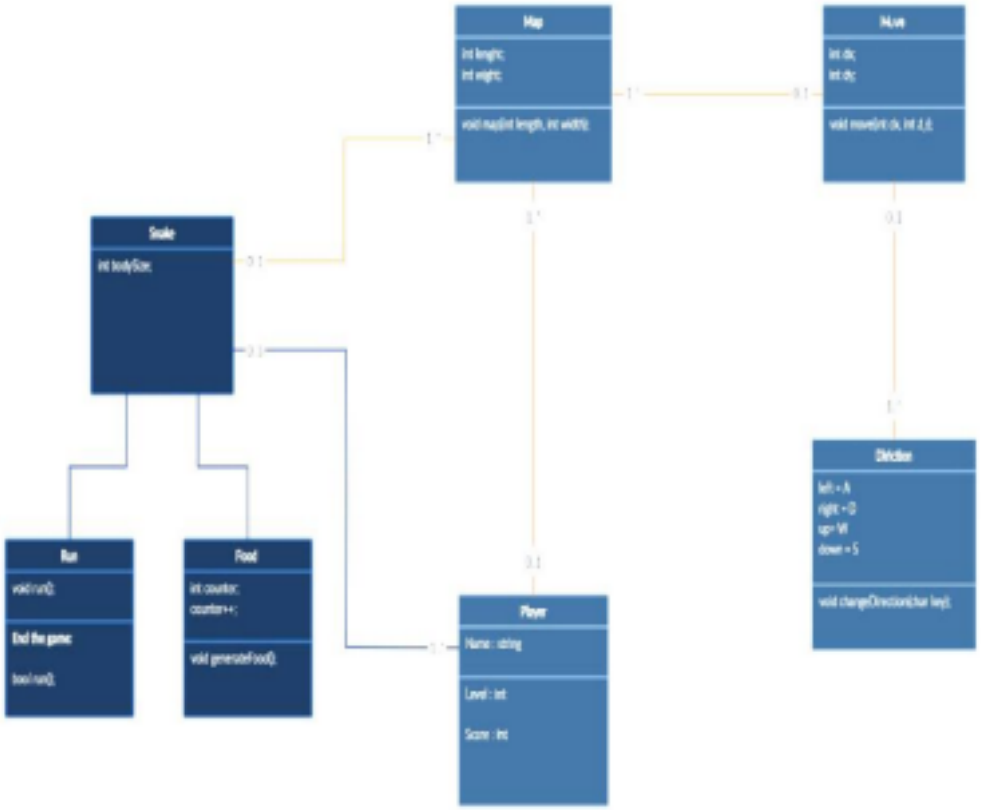
Finally, the **getLength()** and **getBody()** methods return the length and body vector of the snake, respectively.

UML DIAGRAMS:

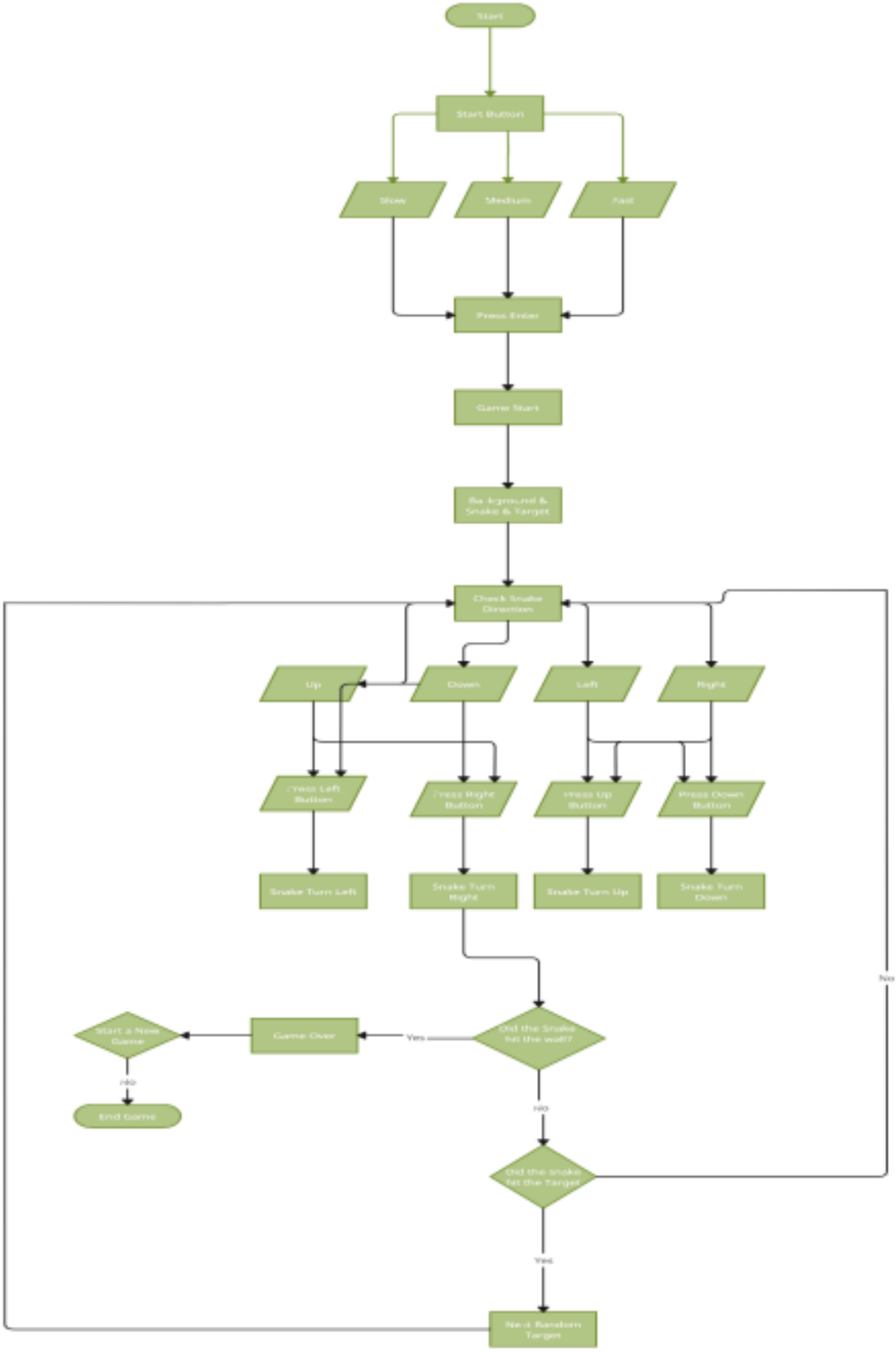
USE CASE DIAGRAM:



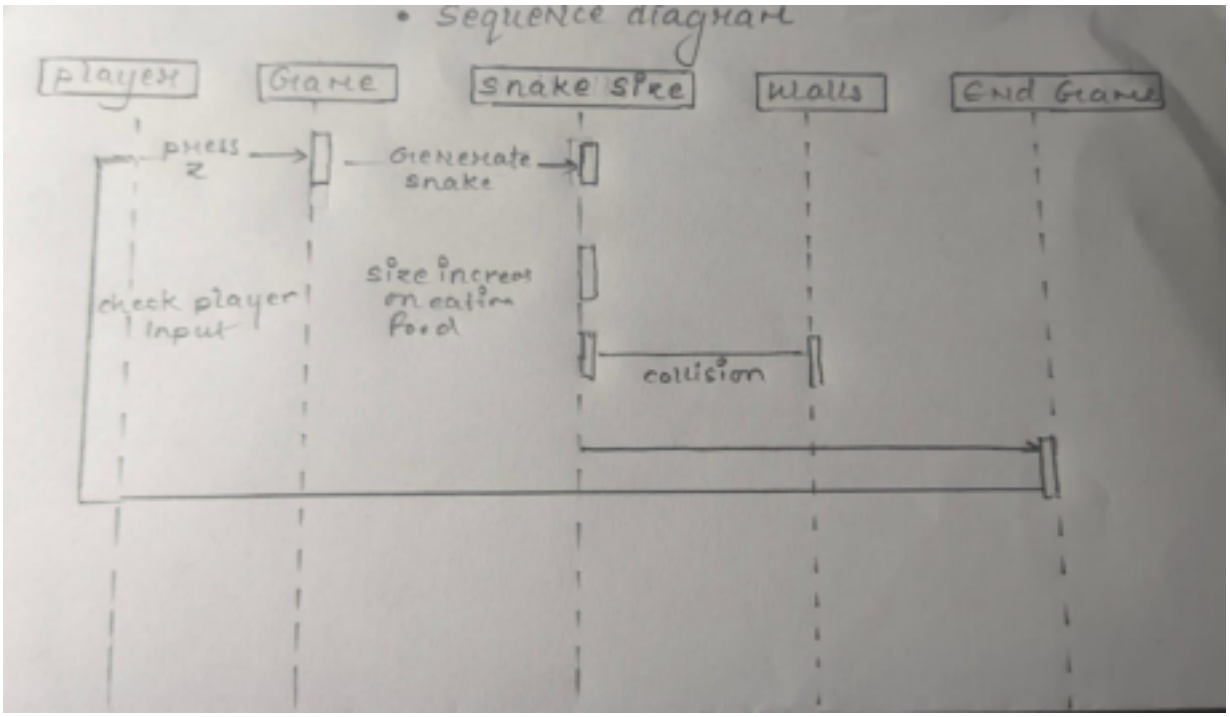
CLASS DIAGRAM:



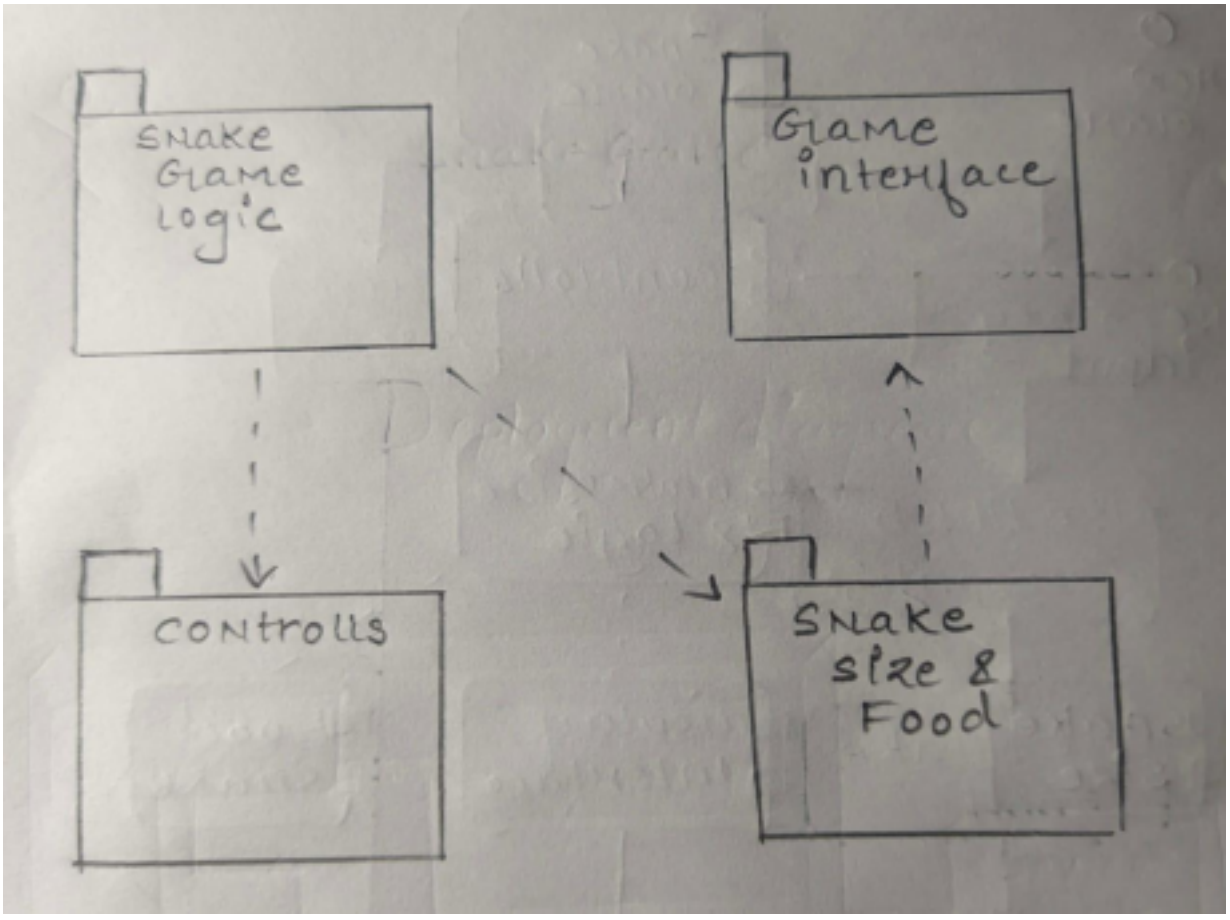
ACTIVITY DIAGRAM:



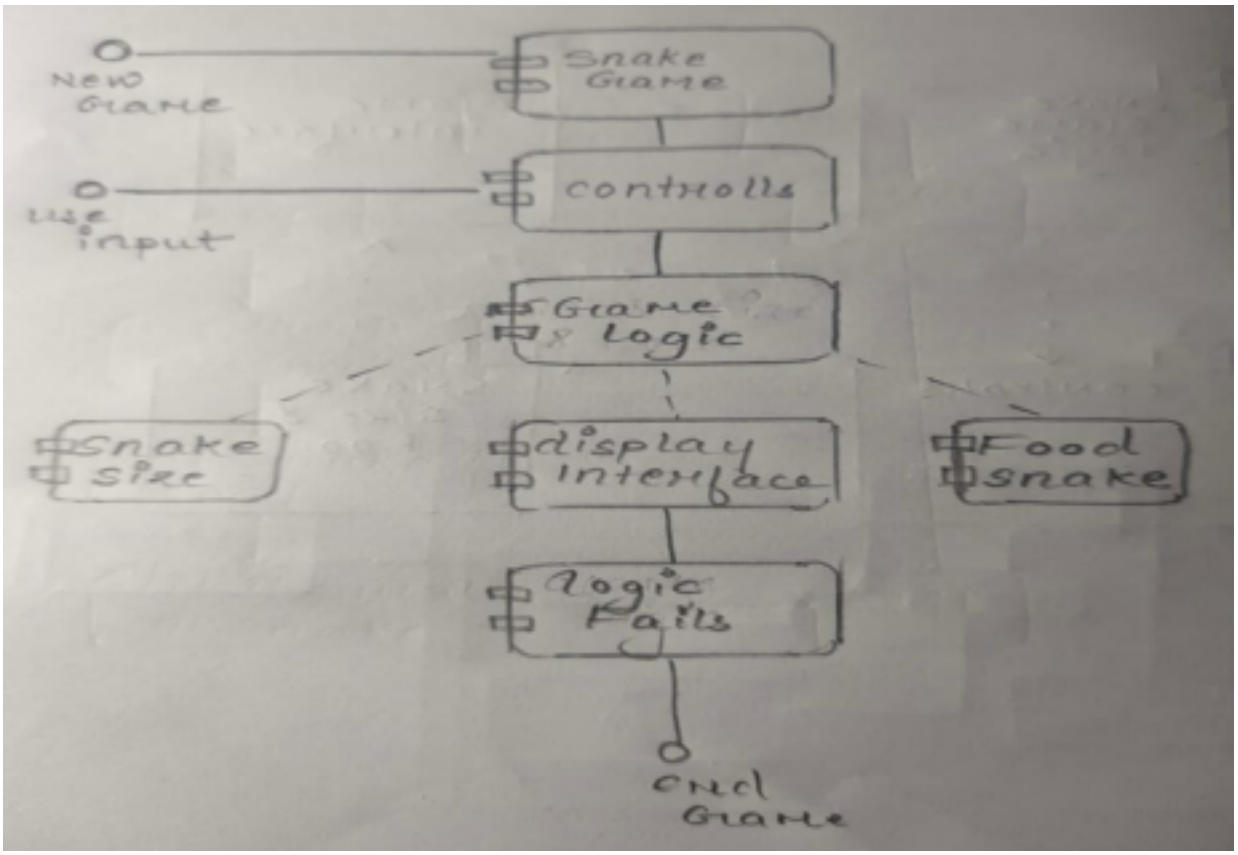
SEQUENCE DIAGRAM:



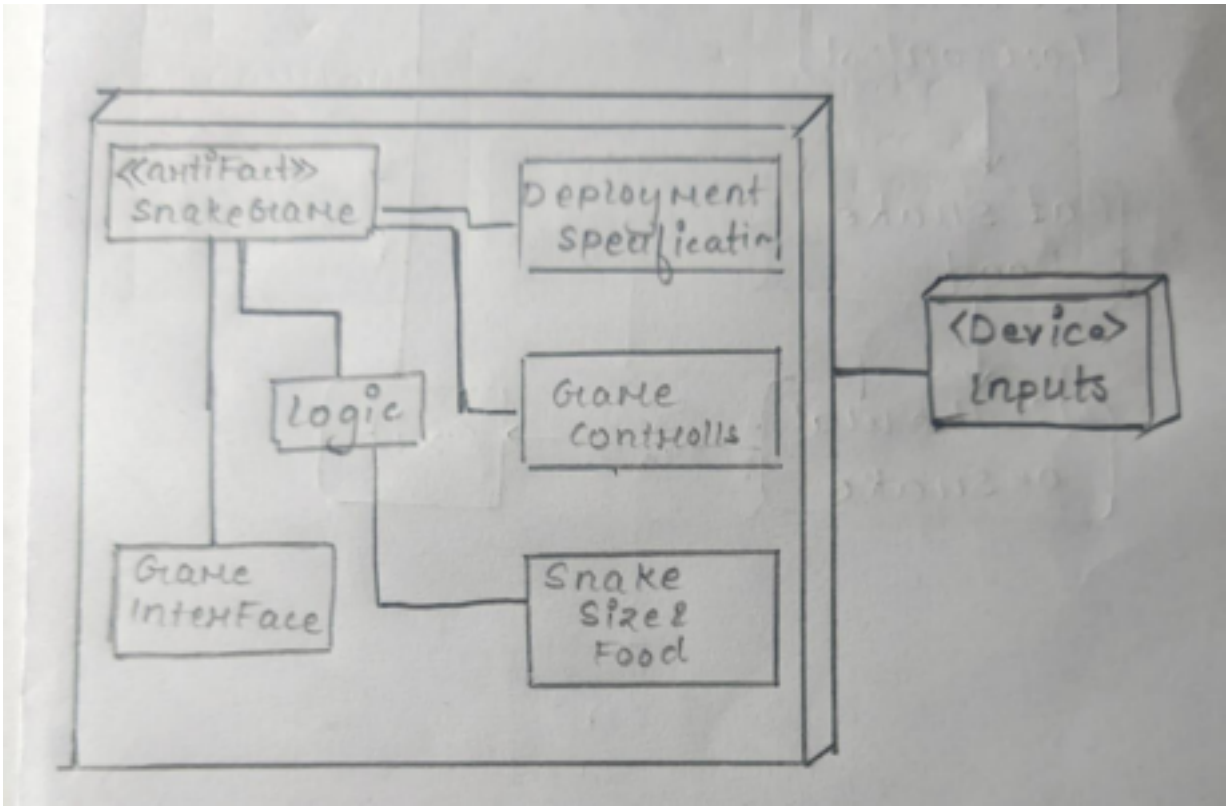
PACKAGE DIAGRAM:



COMPONENT DIAGRAM:



DEPLOYMENT DIAGRAM:



STATE CHART DIAGRAM:

COLLABORATION DIAGRAM:

SOURCE CODE:

```
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <conio.h>
#include <fstream>

using namespace std;
typedefstructtailpos
{
int x;
int y;
structtailpos *next;
structtailpos *prev;
} tail;

int d = 4;

class snake
{
public:
intwallsX, wallsY;
intwalleX, walleY;

int score;

intfoodx, foody;

 HANDLE console_handle;
 COORD cur_cord;
tail *start, *current, *newtail;
snake();
void insert(int x, int y);
void draw();
voiddrawWall();
void move();
bool collision();
voiddrawfood(int x = 0);
voiddrawinit();
voidlabelDead();
void menu();
void help();
};
void loop(snake &ob);

snake::snake()
```

```
{
score = 0;
start = NULL;
current = NULL;
newtail = NULL;
console_handle = GetStdHandle(STD_OUTPUT_HANDLE);
foodx = 12;
foody = 14;
```



```
wallsX = 2;
wallsY = 2;
walleX = 70;
walleY = 20;

cur_cord.X = 152;
cur_cord.Y = 500;
SetConsoleScreenBufferSize(console_handle, cur_cord);
}
void snake ::insert(int x, int y)
{
if (start == NULL)
 {
newtail = new tail;
newtail->x = x;
newtail->y = y;
newtail->next = NULL;
newtail->prev = NULL;
start = newtail;
current = newtail;
 }
else
 {
newtail = new tail;
newtail->x = x;
newtail->y = y;
newtail->next = NULL;
newtail->prev = current;
current->next = newtail;
current = newtail;
 }
}

void snake::move()
{
tail *tmp, *cur;

tmp = current;
while (tmp->prev != NULL)
 {
tmp->x = tmp->prev->x;
tmp->y = tmp->prev->y;
tmp = tmp->prev;
 }
if (d == 1)
start->y--;
```

```
if (d == 2)
start->y++;

if (d == 3)
start->x--;

if (d == 4)
start->x++;
}
```



```
void snake::draw()
{

cur_cord.X=10;
cur_cord.Y=5;

SetConsoleCursorPosition(console_handle,cur_cord);
cout<< "use above statement to to set cursor position";


cur_cord.X = 2;
cur_cord.Y = 0;

SetConsoleCursorPosition(console_handle, cur_cord);
cout<< "SCORE : " << score;

tail *tmp, *last;
tmp = start;
last = current;

while (tmp != NULL)
 {
cur_cord.X = tmp->x;
cur_cord.Y = tmp->y;
SetConsoleCursorPosition(console_handle, cur_cord);
cout<< (char)219;
tmp = tmp->next;
 }
cur_cord.X = last->x;
cur_cord.Y = last->y;
SetConsoleCursorPosition(console_handle, cur_cord);
cout<< ' ';
cur_cord.X = foodx;
cur_cord.Y = foody;
SetConsoleCursorPosition(console_handle, cur_cord);
cout<< (char)15;
}

void snake::drawWall()
{

cur_cord.X = wallsX;
for (int y = wallsY; y <= walleY; y++)
 {
```

```
cur_cord.Y = y;
SetConsoleCursorPosition(console_handle, cur_cord);
cout<< '#';
 }

cur_cord.Y = wallsY;
for (int x = wallsX; x <= walleX; x++)
 {
cur_cord.X = x;
SetConsoleCursorPosition(console_handle, cur_cord);
cout<< '#';
 }
cur_cord.X = walleX;
```



```
for (int y = wallsY; y <= walleY; y++)
 {
cur_cord.Y = y;
SetConsoleCursorPosition(console_handle, cur_cord);
cout<< '#';
 }


cur_cord.Y = walleY;
for (int x = wallsX; x <= walleX; x++)
 {
cur_cord.X = x;
SetConsoleCursorPosition(console_handle, cur_cord);
cout<< '#';
 }
 }

void snake::drawfood(int x)
 {
tail *tmp;
tmp = start;
if (x == 1)
 {
foodx = rand() % 2 + 39;
foody = rand() % 2 + 16;

while (tmp->next != NULL)
 {
if (foodx == tmp->x &&foody == tmp->y)
{
drawfood(1);
cout<< "drawn";
 }

tmp = tmp->next;
 }
 }
 }

void snake::drawinit()
 {
drawWall();
```

```
}
bool snake::collision()
{
tail *tmp;
tmp = start->next;
while (tmp->next != NULL)
 {
if (start->x == tmp->x && start->y == tmp->y)
return true;

tmp = tmp->next;
 }
if (start->x == foodx&& start->y == foody)
{
insert(foodx, foody);
```



```
drawfood(1);
score++; }

for (int x = wallsX; x <= walleX; x++)
 {
if (start->x == x && start->y == wallsY)
{
return true;
 }
 }
for (int y = wallsY; y <= walleY; y++)
 {
if (start->x == wallsX&& start->y == y)
{
return true;
 }
 }

for (int y = wallsY; y <= walleY; y++)
 {
if (start->x == walleX&& start->y == y)
{
return true;
 }
 }
for (int x = wallsX; x <= walleX; x++)
 {
if (start->x == x && start->y == walleY)
{
return true;
 }
 }

return false;
 }
void snake::labelDead()
{

cur_cord.X = (walleX / 2);
cur_cord.Y = (walleY / 2);
```

```
SetConsoleCursorPosition(console_handle, cur_cord);

cout<< "YOU ARE DEAD\n";

cur_cord.Y = (walleY / 2) + 1;
SetConsoleCursorPosition(console_handle, cur_cord);
cout<< "YOUR SCORE IS " << score;
}

void snake::menu()
{
char word;
ifstreamiFile("menu.txt");
word = iFile.get();
while (iFile)
 {
```



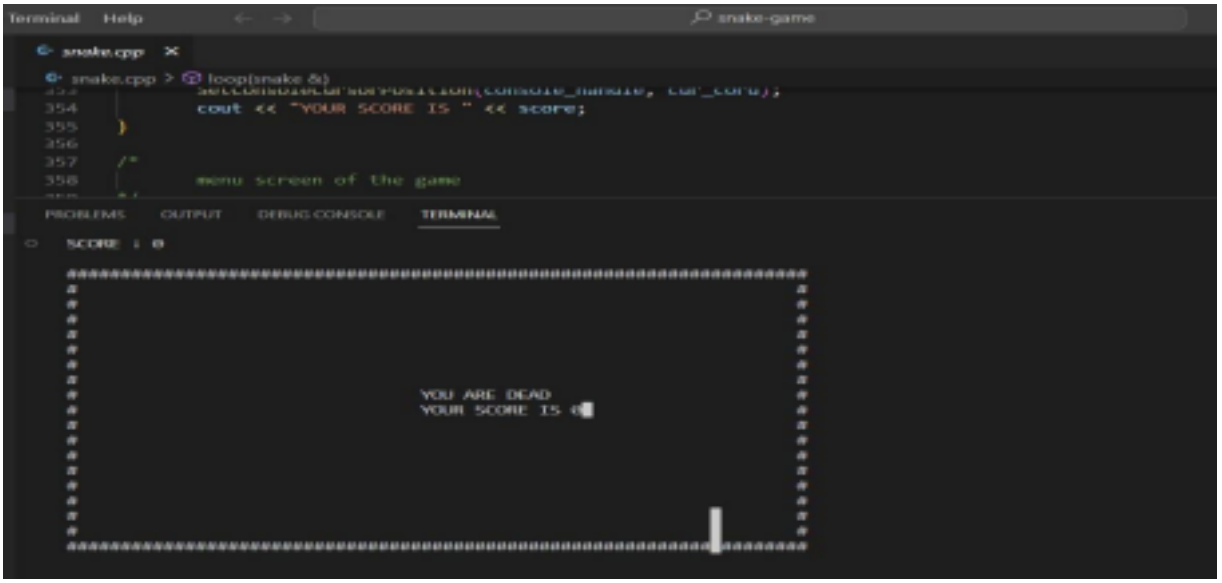
```
cout<< word;
word = iFile.get();
 }
iFile.close();

}

void snake::help()
{
char word;
ifstreamiFile("help.txt");
word = iFile.get();
while (iFile)
 {

cout<< word;
word = iFile.get();
 }
iFile.close();

getch();
}
int main()
{snakeobc;
obc.menu();
switch (getch())
 {
case 'z':
system("CLS");
loop(obc);
break;
case 'h':
system("CLS");
obc.help();
system("CLS");
main();
break;
case 'q':
```

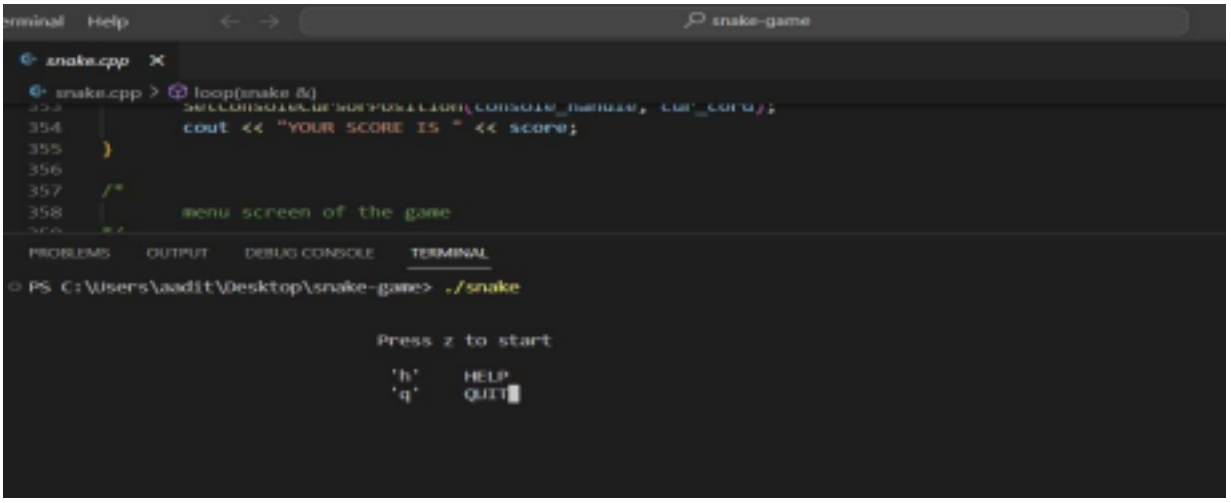
```
break;
default:
system("CLS");
main();
 }

return 0;
 }
void loop(snake &ob)
 {
ob.insert(10, 6);
ob.insert(10, 7);
ob.insert(10, 8);
ob.insert(10, 9);

ob.drawinit();
intdir = 1;
while (1)
```

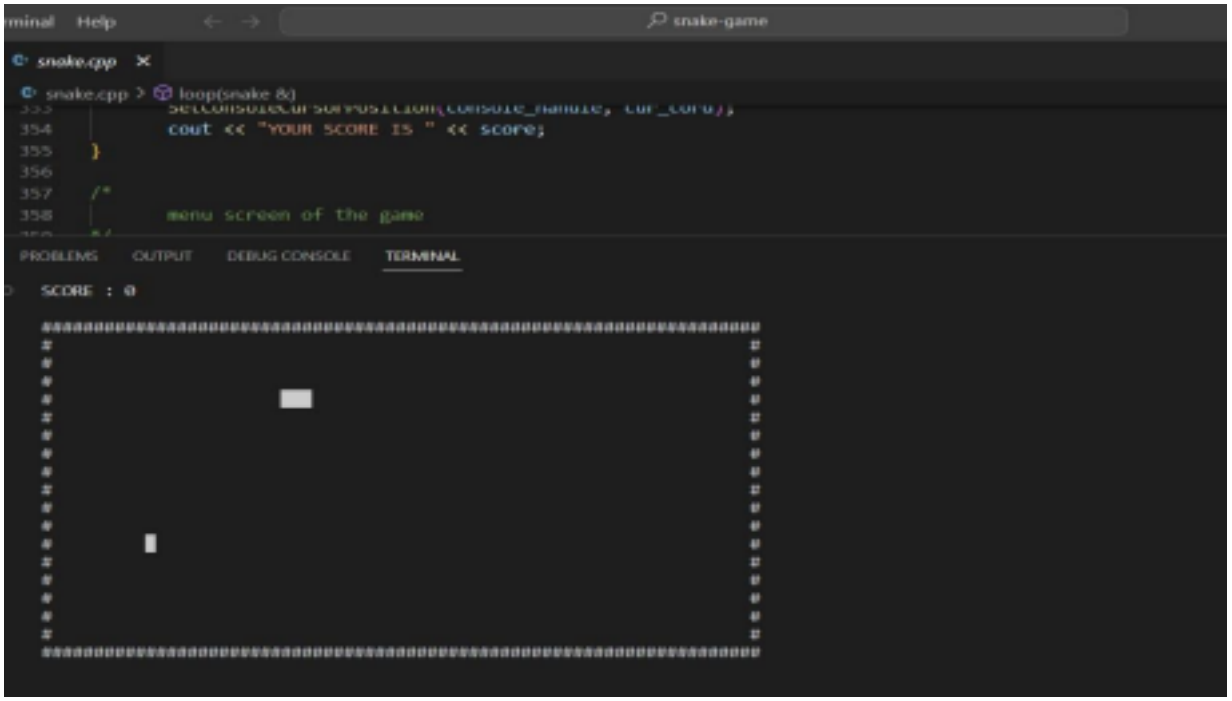


```
 {
ob.draw();
Sleep(200);

if (ob.collision())
{
ob.labelDead();
break;
 }

if (kbhit())
 {
switch (getch())
 {
case 'w':
 d = 1; break;
case 's':
 d = 2; break;
case 'a':
 d = 3; break;
case 'd':
 d = 4; break;
case 'm':
ob.insert(10, 7);
break;
 }
 }

ob.move();
 }
int x;
cin>> x;
 }
```

OUTPUT:


snake-game
snake.cpp
cout << "YOUR SCORE IS " << score;
/*
menu screen of the game
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
SCORE : 0
YOU ARE DEAD
YOUR SCORE IS 0


snake-game
snake.cpp
cout << "YOUR SCORE IS " << score;
/*
menu screen of the game
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\aadit\Desktop\snake-game> ./snake
Press z to start
'h' HELP
'q' QUIT



CONCLUSION AND RESULT:

A snake game was created using C++. The game ran perfectly with no error and was then made to an .exe file for execution on other devices.

REFERENCE:

- Geeksforgeeks.com
- Stackoverflow.com
- Github.com
- Youtube.com