# ABSTRACT

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.

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.

|  |  |
| --- | --- |
| Acknowledgement | i |
| Abstract  Table of contents | ii iii |
| 1. **Introduction**    1. Computer graphics History    2. Application    3. OpenGL History    4. Features of OpenGl Functions of OpenGL    5. Features of OpenGl    6. Functions of OpenGL | **1**  1  2  2  3  4  4 |
| **2 Literature Overview** | **5** |
| 1. **Requirements and specification**    1. Purpose    2. Specific Requirements    3. Functional Requirements    4. Non-Functional Requirements    5. Software requirements Hardware Requirements | **6**  6  7  7  7  8 |
| 1. **System Design & Implementation**    1. Design procedure    2. Control Flow    3. Implementation       1. System Function       2. User defined Function       3. Transformation Functions | **9**  9  10  10  10  12  12 |

|  |  |
| --- | --- |
| 1. **Source Code**    1. Start screen    2. Scenes    3. Main functions | **13**  13  14  16  16 |
| **6 Snapshots** | **23** |
| **7 Conclusion**  Bibliography Appendix | **26**  27  28 |

**LIST OF FIGURES**

1. Fig 1.1.1: CG Flow diagram. 1
2. Fig 1.6.1: Basic opengl structure 4
3. Fig 3.2.1: Basic structure of CG… 8
4. Fig 4.2.1: Control flow diagram… 10
5. Fig 6.1: Start page 26
6. Fig 6.2: Game starting page 26
7. Fig 6.3: Moving towards food 27
8. Fig 6.4: Consuming food 27
9. Fig 6.5: Summary page 28

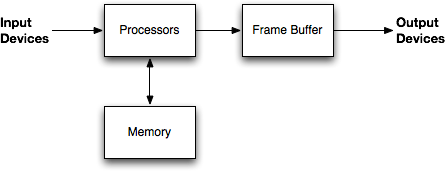
# CHAPTER 1

## INTRODUCTION

* 1. **COMPUTER GRAPHICS**

The dominant characteristics of this new millennium are how computer and communication technologies have become dominant forces in our life. Activities as wide- ranging as film making, publishing, banking and education continue to undergo revolutionary changes as these technologies alter the ways in which we conduct our daily activities. This combination of computers, networks, and the complex human visual system, through computer graphics, has lead to new ways of displaying information, seeing virtual worlds, and communicating with people and machine

* + - Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D Or 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly.
    - Computers have become a powerful medium for the rapid and economical production of pictures.
    - Graphics provide a so natural means of communicating with the computer that they have become widespread.
    - Interactive graphics is the most important means of producing pictures since the invention of photography and television.
    - We can make pictures of not only the real world objects but also of abstract objects such as mathematical surfaces on 4D and of data that have no inherent geometry.
    - A computer graphics system is a computer system with all the components of the general purpose computer system. There are five major elements in system: input devices, processor, memory, frame buffer, output devices.



**Fig1.1.1: CG Flow diagram**

## COMPUTER GRAPHICS HISTORY

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. Computer Graphics become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which Graphical displays cannot be used to some advantage so it is not surprising to find the use of CG so widespread.

Although early application in engineering and science had to relay on expensive and cumbersome equipment’s, advances in computer technology have made interactive computer graphics a practical tool. Today Computer Graphics is found in a diverse area such as science, engineering, medicine, business, industry, art, entertainment, education and training. Computer graphics can be used as a generalized tool for drawing and creating pictures and stimulate the real-world situations within a small computer window. User can create images by computers that are indistinguishable from photographs of real objects.

## APPLICATIONS OF COMPUTER GRAPHICS

Nowadays Computer Graphics used in almost all the areas ranges from science, engineering, medicine, business, industry, government, art, entertainment, education and training.

### CG in the field of CAD:

Computer Aided Design methods are routinely used in the design of buildings, automobiles, aircraft, watercraft, spacecraft computers, textiles and many other applications.

### CG in presentation Graphics:

Another major application area presentation graphics used to produce illustrations for reports or generate slides. Presentation graphics is commonly used to summarize financial, statistical, mathematical, scientific data for research reports and other types of reports.2D and 3D bar chart to illustrate some mathematical or statistical report.

### CG in computer Art:

CG methods are widely used in both fine art and commercial art applications. Artists use a variety of computer methods including special purpose hardware, artist’s paintbrush program (lumen), other pain packages, desktop packages, maths packages, animation packages that provide facility for designing object motion. Ex: cartoons decision is an example of computer art which uses CG.

### Entertainment

Computer graphics methods are now commonly used in making motion pictures, music, videos, games and sounds. Sometimes graphics objects are combined with the actors and live scenes.

### Education and Training

Computer generated models of physical financial, economic system is often as education aids. For some training application special systems are designed. Ex: specialized system is simulator for practice sessions or training of ship captain, aircraft pilots and traffic control.

### Image Processing

Although the methods used in CG image processing overlap, the 2 areas are concerned with fundamentally different operations. In CG a computer is used to create picture. Image processing on the other hand applies techniques to modify existing pictures such as photo scans, TV scans.

### User Interface

It is common for software packages to provide a graphical interface. A major component of a graphical interface is a window manager that allows a user to display multiple window area. Interface also displays menus, icons for fast selection and processing.

### HISTORY OF OPENGL

William fetter was credited with coining the term Computer Graphics in 1960, to describe his work at Boeng. One of the first displays of computer animation was future world (1976), which included an animation of a human face and hand-produced by Car mull and Fred Parkle at University of Utah.

There are several international conferences and journals where the most significant results in computer graphics are published. Among them are the SIGGRAPH and Euro graphics conferences and the Association for computing machinery (ACM) transaction on Graphics journals.

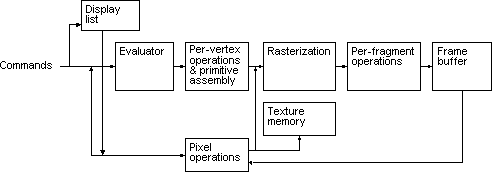
### FEATURES OF OPENGL:

The main features of OpenGL are

* + - It provides 3D geometric objects such as lines, polygons, triangle, meshes, spheres, cubes, quadric surface, curved surfaces.
    - It provides 3D modeling transformations and viewing functions to create views of 3D scenes using the idea of a virtual camera
    - It supports high-quality rendering of scenes, including hidden-surface removal, multiple light sources, material types, transparency, textures, blending, fog.

### FUNCTIONS OF OPENGL:

The below figure shows the organization of the libraries for an X Window System environment. For this window system, GLUT will use GLX and the X libraries. The application program, however, can use only GLUT functions and thus can be recompiled with the GLUT library for other window systems.



**Fig 1.6.1: Basic opengl structure**

# CHAPTER 2

## LITERATURE OVERVIEW

The basic functions like glcolor3f(…..); glrotatef(…..); gltranslate(…..) etc. that are most commoly Used in the code are taken form the prescribed VTU text book “INTERACTIVE COMPUTER GRAPHICS” 5th edition by Edward Angel.

The lab programs in the syllabus also serve as a basic template for creating a project. The usage of colors and specifications are taken from the various programs that were taught in the lab. The VTU prescribed text book serves as a huge database of functions and they are used in the project.

The C concepts which are used, are being taken from the book named Yeshwant Kanitkar. Some concepts like constructing bowl and fountain are taken from the search results in the codecolony.com.

The purpose of the References and Reviews Department is to collect and to

disseminate rapidly as many publication citations of interest as come to the attention of

the editor. Areas of interest cover all aspects of Computer Graphics and Interactive

Techniques, particularly the following: man-machine interaction, graphical

representation of data, data structures, data base searching and retrieval, computer aided

design, graphic control systems, network design, computer art, computer animation,

graphic software, graphic hardware, and the areas listed in the taxonomy by the GSPC

Subcommittee .Technical reports, Master's and Ph.D. these will be included in the list of

current literature, provided that copies of these will be generally available, either on

request, through interlibrary loan, on microfilm, microfiche or other means. Such

references should be submitted to the departmental editor .

# CHAPTER 3

## REQUIREMENTS AND SPECIFICATIONS

### Purpose of the Requirements Document:

The software requirement specification is the official statement of what is required for development of particular project. It includes both user requirements and system requirements. This requirement document is utilized by variety of users starting from project manager who gives project to the engineer responsible for development of project.

It should give details of how to maintain, test, verify and what all the actions to be carried out through life cycle of project.

### Scope of the Project

The scope is to use the basic primitives defined in openGL library creating complex objects. We make use of different concepts such as pushmatrix(),translate() ,popmatrix(),timer function.

### Definition

The project ***HUNGRY SNAKESS*** is created to demonstrate OpenGL’s concepts. It encompasses some of the skills learnt in our OpenGL classes such as pushmatrix(),translate()

,popmatrix(),timer function.

### Acronyms & Abbreviations

OpenGL provides a powerful but primitive set of rendering command, and all higher level design must be done in terms of these commands.

OpenGL Utility Toolkit(GLUT) :- windows-system-independent toolkit.

### References

OpenGL tutorials

Interactive Computer Graphics(Edward Angel)

## Specific requirements

### USER REQUIREMENT:

User requirements in project management should be specified in terms of two categories.

They are as follows.

* + 1. Functional requirements
    2. Non-functional requirement

### FUNCTIONAL REQIREMENT

* Software should have proper, easy to use interface.
* Software should be user friendly and it should not take much time for the user to learn how to use.
* It should be possible to draw any geometric shapes like lines, triangles, polygon in an easy way.
* It should provide at least these colors for usage. Red, Green, Blue, Yellow, White, Magenta, Black.
* It should provide white background to draw objects and pick buttons to pick the tools.

### NON- FUNCTIONAL REQIREMENT

These are the constraints on the services of functions offered by the system, they include timing constraints, constraints on the development process and standards and processes. They are:

* The software should be easy to install into the computer.
* It should be very fast and no delay should be present in its operation.
* It should not cause system hanging during any error condition.
* It should be tolerant to user errors.
* It should not create any security problems for the computer. It should not overload the system Easy to understand and should be simple.
* OpenGL library facilities should be used.

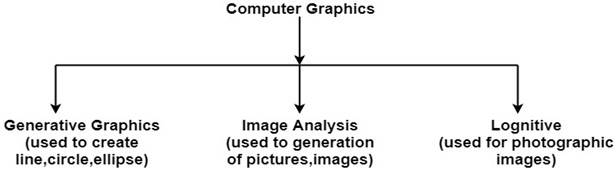
## Software Requirements:

In this section the various requirements that are essential for this project are specified. These requirements have to be fulfilled for execution of the project. The purpose, scope along with hardware and software requirements are given below.

* Ubuntu OS.
* Eclipse compiler.

## Hardware Requirements:

* Processor- Intel or AMD(Advanced Micro Devices)
* RAM- 512MB(minimum)
* Hard Disk-1MB(minimum)
* Mouse
* Keyboard
* Monitor



**Fig 3.2.1: Basic structure of Computer Graphics**

# CHAPTER 4

## SYSTEM DESIGN AND IMPLEMENTION

The software consists of a main program from which the control starts, and a number of other functions for displaying, initializing and other purposes. The architecture of the software is explained below.

* + The control starts from the main program; the software initializes some variables and environments and then enters the event driven loop.

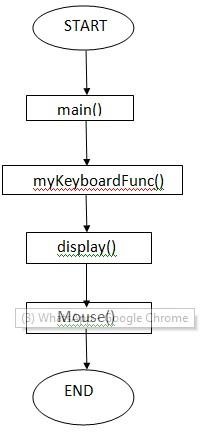
## DESIGN PROCEDURE

The Game is presented by using various Standard openGL and user defined functions. The following list gives an overview of the functions used and an implementation section defines how actually the project is implemented.

**Rules to Operate:**

Our project execution will starts from the main.

* + - When the user compile the program it will create the start screen.
    - In that window the title of the project and the names of the students are displayed.
    - Then press 1 to get the scene displayed in a separate window.
    - Here the convoy vehicle starts moving and then press 1 ,the suicide bomber vehicle starts moving in opposite direction.
    - After the explosion , it moves to the next screen in the separate window.
    - User can press 2 to go back to the previous screen.
    - Then user has to press the mouse right button.
    - Then the user has to choose the first option to light the candles that will be displayed in the window.
    - Finally ,the user can select the quit option to exit from the screen.
  1. **CONTROL FLOW DIAGRAM**



**Fig 4.2.1: Control flow diagram**

## IMPLEMENTATION

* + 1. ***SYSTEM FUNCTIONS***
       - **Stdio.h:** All functions in C (and its many derivatives) are declared in header files thus, programmers have to include the stdio.h header in the source code in order to use the functions declared in it.
       - **Stdlib.h:** Stdlib.h is the header of the purpose standard library of C programming language which includes functions involving memory allocation, process control, conversions and others. The name “stdlib.h” stands for standard library .
       - **GL/glut.h:** It includes the definitions for the macro such as GL\_POINTS, GL\_LINES, and GL\_POLYGON etc, which are used by the programmer.
       - **glClear(GL\_COLOR\_BUFFER\_BIT**)**:** The glClear() function clears a particular buffer or combination of buffers. A buffer is a storage area for image information. The red, green, and blue components of a drawing actually have separate buffers, but they are usually collectively referred to as the color.
       - **glBegin(glEnum mode):** Initiates the new primitive of the type mode and starts the collection of vertices. Values of mode include GL\_POLYGON, GL\_POINTS, etc.,
       - **glMatrixMode( ):** Specifies the current matrix (PROJECTION, MODELVIEW, TEXTURE).
       - **glLoadIdentity( ):** Sets the current matrix to identity.
       - **glutPostRedisplay( ):** Request that the display callback be executed after the current windows.
       - **glutDisplayFunc( ):** glutDisplayFunc sets the display callback for the current window.
       - **glutMainLoop( ):** After the GLUT program has done initial setup such as creating windows and menus, GLUT programs enter the GLUT event processing loop by calling glutMainLoop.
       - **glutInit( ):** glutInit is used to initialize the GLUT library.
       - **glutReshapeFunc(void \*f(int width,int height )):** Registers the reshape callback f. The callback function returns the height and width of the new window. The reshape callback invokes display callback.
       - **glutIdleFunc(void (\*f)(void)):** Registers the display callback function f i.e., executed whenever there are no other events to be handled.
       - **glutSwapBuffers( ):** Swaps the front and back end.

### gluOrtho2D(GLdouble left,GLdouble right,GLdouble bottom,GLdouble top):

Defines a twodimensional viewing rectangle in the plane Z=0.

* + - * **Void glutInit(int \*\*argc,char \*\*argv):** Initializes GLUT.
      * **Void glutInitWindowSize(int width, int height):** Specifies the initial height and width of windows in pixel.
      * **Void glutInitDisplayMode(unsigned int mode):** Request a display with the properties in mode.
      * **Int glutCreateWindow(char \*title):** Creates a window on the display.
      * **Void glFlush( ):** Forces any buffered OpenGL commands to execute.
      * **Void glutDisplayFunc(void (\*func)(void)):** Register the display function that is executed when the window needs to be redrawn.
      * **Void glutBitmapCharacter(void \*font, int char):** Renders a character with ASCII code char at the current raster position using the raster font given by font.

## USER DEFINED FUNCTIONS

### myKeyboardFunc():

It is used to give input to the game.

### void coloroct():

It is used to coloring the octate.

### void color\_score():

It is used to coloring the score box.

### void pointer() :

It is used to indicate the score of the player.

### void mouse() :

It is used to control the rotation of octate.

### void spinsocta() :

It is used for rotation of octate.

### void scoreboard1():

It is used to display the palyer1 score.

* + 1. **Transformation Functions**

**glTranslate Function:** The glTranslated and glTranslatef functions multiply the current matrix by a translation matrix.

**SYNTAX:** void glTranslate( x, y, z);

**PARAMETERS:** x, y, z - The x, y, and z coordinates of a translation vector.

## Funtions used to display

**glMatrixMode Function:** The glMatrixMode function specifies which matrix is the current matrix.

**SYNTAX:** void glMatrixMode(GLenum mode);

**PARAMETERS:** Mode- The matrix stack that is the target for subsequent matrix operations. The mode parameter can assume one of three values:

**glLoadIdentity Function :** The glLoadIdentity function replaces the current matrix with the identity matrix.

**SYNTAX:** void glLoadIdentity(void);

# CHAPTER 5

# SOURCE CODE

**FUNCTIONS:**

A function is a block of code that has a name and it has a property that it is reusable that is it can be executed from as many different points in a c program as required. The partial code of various function that have been used in the program are:

### Start screen

#include "StdAfx.h"

#include <stdio.h>

#include <time.h>

#include <windows.h>

#include <gl/glut.h>

#define UP 1

#define Down 2

#define LEFT 3

#define RIGHT 4

// Status Variables

GLint lvl = 1;

GLint points = 0;

GLint size = 0;

GLbyte gameOver = true;

GLbyte EnableLight = true;

// Snake Variables

GLint bodyPos[2][100] = {{}};

GLint \_x = 5;

GLint \_z = 10;

GLint \_oldX[2] = {};

GLint \_oldZ[2] = {};

GLbyte direction = 0;

// Food Variables

GLint \_bx = 0;

GLint \_bz = 0;

// Screen variables

GLint \_w = 800;

GLint \_h = 550;

GLint \_Giw = 0;

GLint \_Gih = 0;

GLint \_Gfw = 150;

GLint \_Gfh = 150;

//Variables for the Camera Angle

static GLfloat view\_rotx=45.0F ;

static GLfloat view\_roty=0.0F ;

static GLfloat view\_rotz=0.0F ;

static GLfloat headRotation=90.0F ;

static GLfloat zoom=-300.0f;

//Configure the lightning

void initLight()

{

//Add ambient light

GLfloat ambientColor[] = {0.3f, 0.4f, 0.8f, 1.0f};

glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambientColor);

//Add positioned light

GLfloat lightColor0[] = {0.5f, 0.5f, 0.5f, 1.0f};

GLfloat lightPos0[] = {4.0f, 0.0f, 8.0f, 1.0f};

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, lightColor0);

glLightfv(GL\_LIGHT0, GL\_POSITION, lightPos0);

//Add directed light

GLfloat lightColor1[] = {0.5f, 0.2f, 0.2f, 1.0f};

//Coming from the direction (-1, 0.5, 0.5)

GLfloat lightPos1[] = {-1.0f, 0.5f, 0.5f, 0.0f};

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, lightColor1);

glLightfv(GL\_LIGHT1, GL\_POSITION, lightPos1);

}

//initialize the first configurations

void Initialize(void)

{

glEnable(GL\_DEPTH\_TEST);

glClearColor(0.4f, 0.8f, 1.0f, 1.0f); //Change the background to sky blue

if(EnableLight){

glEnable(GL\_COLOR\_MATERIAL); //Enable color

glEnable(GL\_LIGHTING); //Enable lighting

glEnable(GL\_LIGHT0); //Enable light #0

glEnable(GL\_LIGHT1); //Enable light #1

glEnable(GL\_NORMALIZE); //Automatically normalize normals

}

}

//Configure window resize

void resize (int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(45.0, (double)w / (double)h,1, 800.0);

}

void Write(char \*string){//Write string on the screen

while(\*string)

glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, \*string++);

}

//This Function will rotate the object according to the Angles

void ManipulateViewAngle() {

glRotatef(view\_rotx,1.0,0.0,0.0);//Rotate Arround X axis

glRotatef(view\_roty,0.0,1.0,0.0);//Rotate Arround Y axis

glRotatef(view\_rotz,0.0,0.0,1.0);//Rotate Arround Z axis

}

//This Function will reset the snake size and location...

void Reset(){

\_x = 5;

\_z = 10;

direction = 0;

lvl = 1;

points = 0;

size = 0;

gameOver = false;

view\_rotx=45.0F ;

view\_roty=0.0F ;

view\_rotz=0.0F ;

headRotation=90.0F ;

}

//Display a welcome screen

void WelcomeScreen(){

char tmp\_str[40];

glRasterPos2f(-40, 20);

Write("=========WELCOME TO Azzonika Tutorials=========");

glRasterPos2f(-6, 10);

Write("=========Simulation of Snake Game=========");

glRasterPos2f(-60, 0);

Write("=========To Start Playing please press 'S' or 's' =========");

}

void DrawSnake(){

int i;

//Drawing the head & the plane

glPushMatrix();

ManipulateViewAngle();

//This will draw the plane that the snake will run on.

glPushMatrix();

//glColor3f(1, 0.7, 0.2);

glColor3f(0.0, 0.6, 0.2);

glTranslatef(75.0, -16.0, 75.0);

glScalef(155,5.0,155);

glutSolidCube(1);

glPopMatrix();

//Here we will draw the Head of the snake

glColor3f(1,0,0);//Color it red

glTranslatef(\_x,-10.0,\_z);//Give it the location according to \_x & \_z

glScalef(0.5,0.5,0.5);

glutSolidSphere(10,20,20);//Draw the head as a sphere a litle bit bigger than the body spheres

glRotatef(headRotation, 0.0, 1.0, 0.0);

glColor3f(1,0,0);

glTranslatef(0,0.0,6.0);

glScalef(0.8,1.0,1.0);

glutSolidCone(10,10,20,20);

glColor3f(1,1,1);

glTranslatef(-4.0,10.0,0.0);

glScalef(0.3,0.3,0.3);

glutSolidSphere(10,20,20);

glTranslatef(26.0,0.0,0.0);

glutSolidSphere(10,20,20);

glPopMatrix();

//Drawing the body

for(i=0; i<size; i++){//Loop throw the size and draw spheres representing the body

glPushMatrix();

ManipulateViewAngle();

glTranslatef(bodyPos[0][i],-10.0,bodyPos[1][i]);//this will locate the spheres

glColor3f(1,0,0);//Color Red

glScalef(0.5,0.5,0.5);

glutSolidSphere(7,20,20);

glPopMatrix();

}

}

void DrawFood()

{

//Draw the Sphere representing the Food for the snake

glPushMatrix();

ManipulateViewAngle();

glTranslatef(\_bx,-10.0,\_bz);

glColor3f(1, 1, 0);

glScalef(0.5,0.5,0.5);

glutSolidSphere(7,20,20);

glPopMatrix();

}

void GameStatus(){

char tmp\_str[40];

glColor3f(0.8, 0.2, 0);

glRasterPos2f(5, 20);

sprintf(tmp\_str, "Snake Game");

Write(tmp\_str);

// Print the status of the game on the screen

glColor3f(0, 0, 0);

glRasterPos2f(5, 10);

sprintf(tmp\_str, "Level: %d Points: %d", lvl, points);

Write(tmp\_str);

}

//Generates Random Numbers for the location of the food that the snake will eat

int RandomNumber(int high, int low)

{

return (rand() % (high-low))+low;

}

//Generate the New Food that the snake will eat

void newFood(){

time\_t seconds;

time(&seconds);

srand((unsigned int) seconds);

\_bx = RandomNumber(\_Gfw-\_Giw, \_Giw+10);

\_bz = RandomNumber(\_Gfh-\_Gih, \_Gih+10);

}

//This Function Will Check for Collision

bool collision(){

int i;

for(i=0; i<size; i++){

if((bodyPos[0][i] == \_x && bodyPos[1][i] == \_z) ||

//((bodyPos[0][i] >= \_x) && (bodyPos[0][i] <= \_x + 5) && (bodyPos[1][i] >= \_z) && (bodyPos[1][i] <= \_z + 5)) ||

//((bodyPos[0][i] <= \_x) && (bodyPos[0][i] >= \_x - 5) && (bodyPos[1][i] <= \_z) && (bodyPos[1][i] >= \_z - 5)) ||

//((bodyPos[0][i] <= \_x) && (bodyPos[0][i] >= \_x - 5) && (bodyPos[1][i] >= \_z) && (bodyPos[1][i] <= \_z + 5)) ||

((bodyPos[0][i] >= \_x) && (bodyPos[0][i] <= \_x + 5) && (bodyPos[1][i] <= \_z) && (bodyPos[1][i] >= \_z - 5)))

return true;

}

return false;

}

//This Function will move the snake according to the directions

//Taken from the Keyboard keys

void Run(int value){

int i;

\_oldX[1] = \_x;

\_oldZ[1] = \_z;

switch(direction){

case RIGHT:

headRotation =90;

\_x += 6;

if(\_x > \_Gfw-2) \_x = \_Giw-1;//This will check if the snake is going into the border so it will appear on the other side

break;

case LEFT:

headRotation =-90;

\_x -= 6;

if(\_x < 0) \_x = \_Gfw-2;//This will check if the snake is going into the border so it will appear on the other side

break;

case UP:

headRotation =0;

\_z += 6;

if(\_z > \_Gfh-2) \_z = \_Gih-1;//This will check if the snake is going into the border so it will appear on the other side

break;

case Down:

headRotation =180;

\_z -= 6;

if(\_z < 2) \_z = \_Gfh-2;//This will check if the snake is going into the border so it will appear on the other side

break;

}

//Checks for Collisoin if yes Game Over

if(collision()) gameOver = true;

//Checks if the snake ate the food (check the X and Y)

// If yes it will increase the points & the size of the snake & create a new food

if((\_x == \_bx && \_z == \_bz) ||

((\_x >= \_bx) && (\_x <= \_bx + 4) && (\_z >= \_bz) && (\_z <= \_bz + 4)) ||

((\_x <= \_bx) && (\_x >= \_bx - 4) && (\_z <= \_bz) && (\_z >= \_bz - 4)) ||

((\_x <= \_bx) && (\_x >= \_bx - 4) && (\_z >= \_bz) && (\_z <= \_bz + 4)) ||

((\_x >= \_bx) && (\_x <= \_bx + 4) && (\_z <= \_bz) && (\_z >= \_bz - 4))){

points++;

if(points < 100) size++;

if(points%5 == 0 && lvl < 15) lvl++;

newFood();

}

for(i = 0; i<size; i++){// Save the positions of the body parts

\_oldX[0] = \_oldX[1];

\_oldZ[0] = \_oldZ[1];

\_oldX[1] = bodyPos[0][i];

\_oldZ[1] = bodyPos[1][i];

bodyPos[0][i] = \_oldX[0];

bodyPos[1][i] = \_oldZ[0];

}

//Set the Timer

glutTimerFunc(130, Run, 0);

}

void Display(void){//Draw Function

// Clear screen

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

if(EnableLight) initLight();

glTranslatef (-60.0, 40.0, zoom);

//Check if the value of the Flag "Game Over is not True it will continue the game"

if(!gameOver){

GameStatus();

DrawFood();

DrawSnake();

}

else

WelcomeScreen();

// Updates the screen

glutPostRedisplay();

glutSwapBuffers();

}

void Special(int key, int x, int y){

switch(key){

case GLUT\_KEY\_RIGHT :

if(direction != LEFT)

direction = RIGHT;

break;

case GLUT\_KEY\_LEFT :

if(direction != RIGHT)

direction = LEFT;

break;

case GLUT\_KEY\_UP :

if(direction != UP)

direction = Down;

break;

case GLUT\_KEY\_DOWN :

if(direction != Down)

direction = UP;

break;

}

}

void keyboard (unsigned char key, int x, int y)

{

switch (key) {

case 'S' : Reset() ;

glutPostRedisplay();

break;

case 's' : Reset() ;

glutPostRedisplay();

break;

//ESC to Exit

case 27:

exit(0);

break;

default:

break;

}

}

int main(int argc, char\*\* argv) {

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowSize(\_w,\_h);

glutInitWindowPosition(80,80);

glutCreateWindow("Snake Game");

Initialize();

glutSpecialFunc(Special);

glutKeyboardFunc(keyboard);

glutDisplayFunc(Display);

glutReshapeFunc(resize);

newFood();

Run(0);

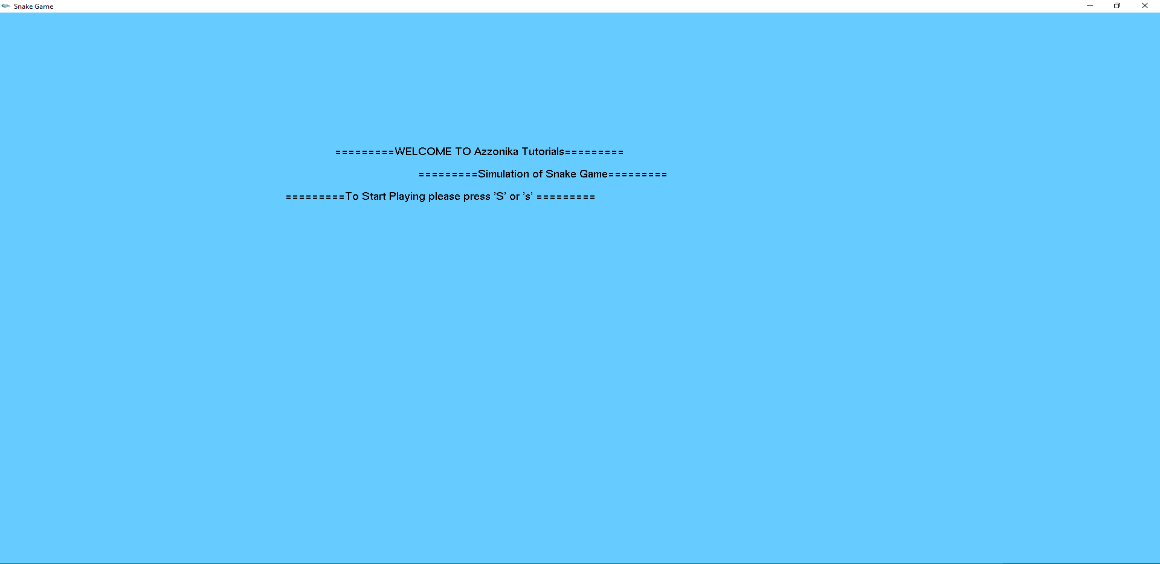
glutMainLoop();

}

# CHAPTER 6

* 1. **START PAGE**

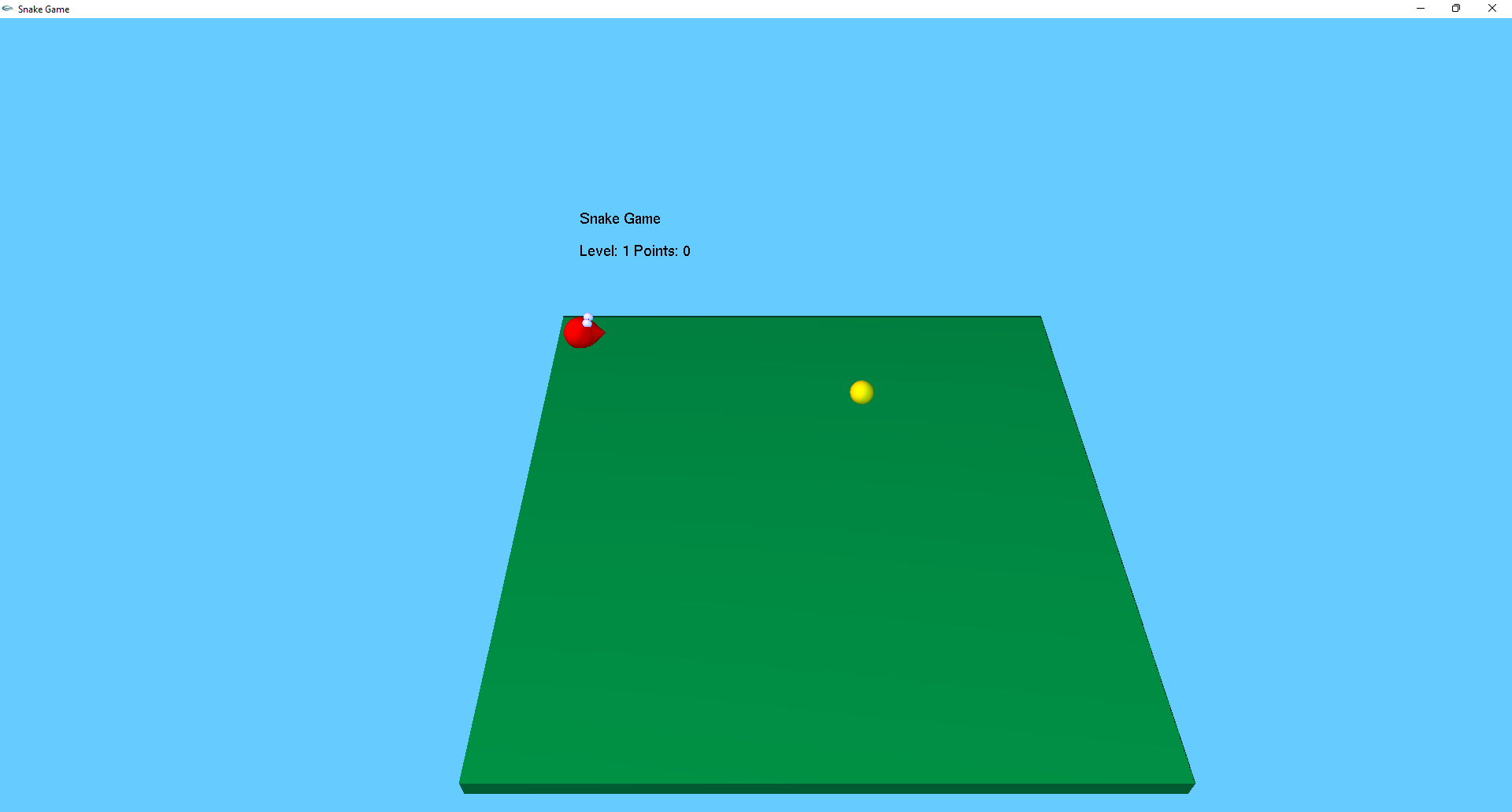
**SNAPSHOTS**



**Fig 6.1: Start page**

As shown on the snapshot this page contains the title of the project and press S to start .

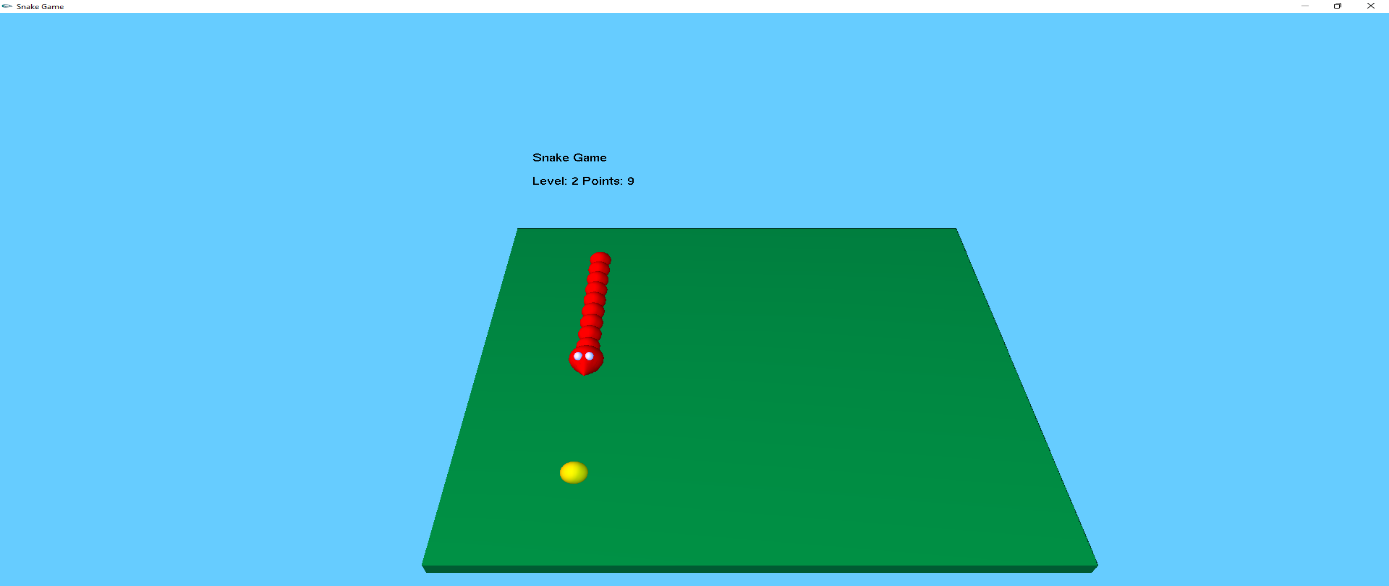
* 1. **GAME STARTING PAGE**



**Fig 6.2: Game starting page**

This page says that starting point of game.

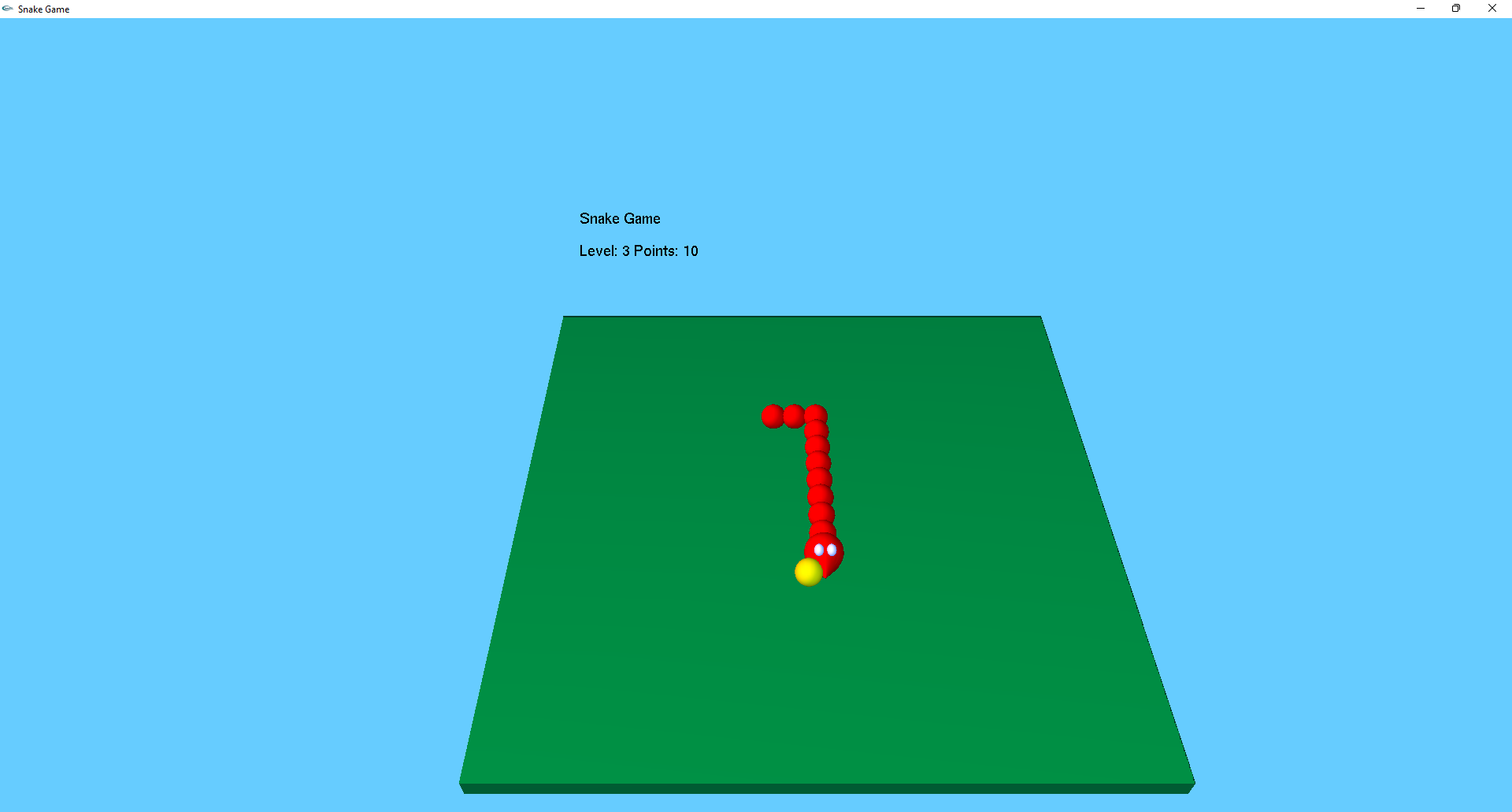
* 1. **SNAKE MOVING TOWARDS ITS’S FOOD**



**Fig 6.3: Snake moving towards it’s food**

This page says that when user press S,then snake starts moving towards it’s food & the snake turning can controlled by arrow keys.

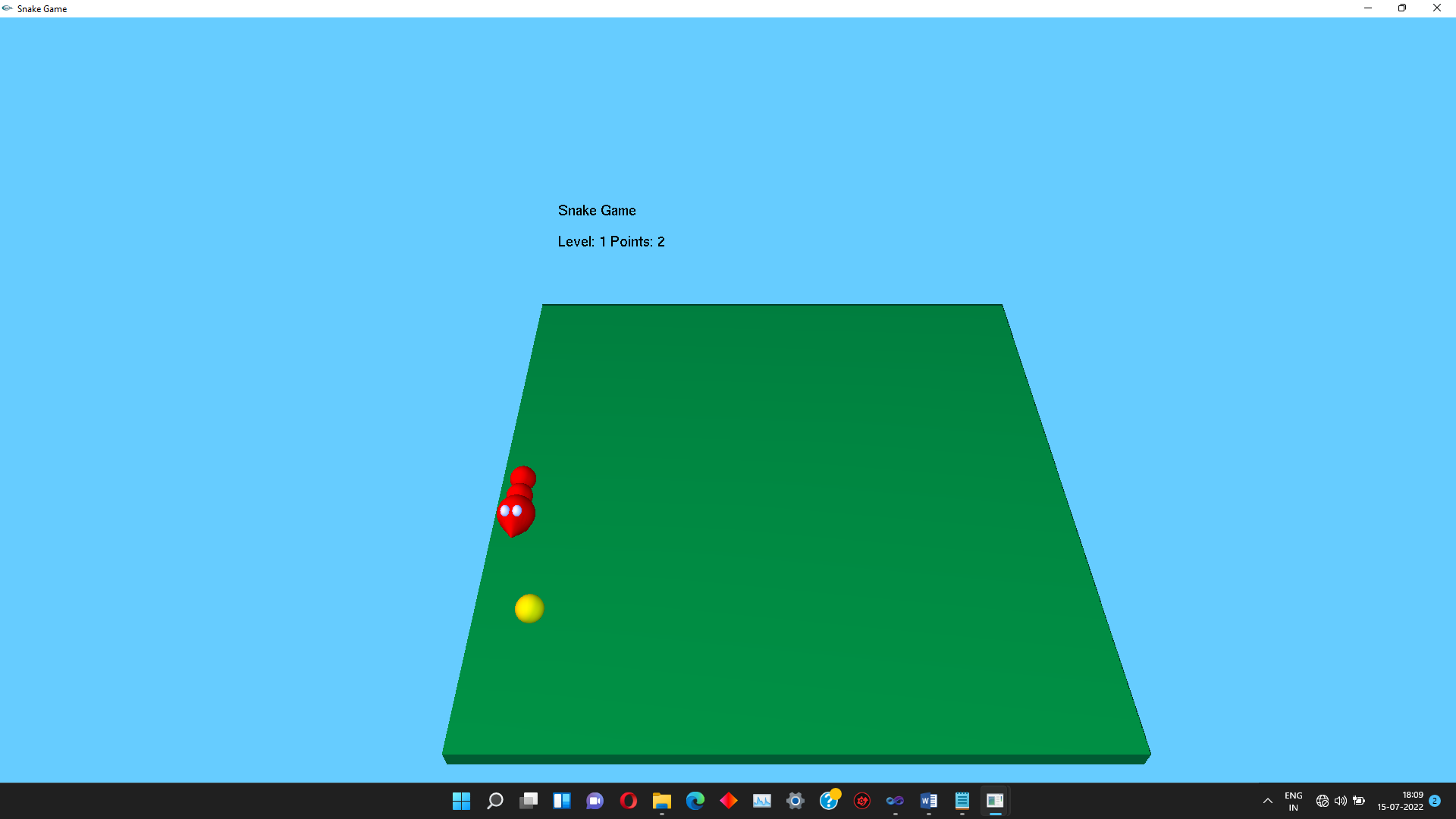
* 1. **CONSUMING FOOD**



**Fig 6.4: Consuming food**

This page says that snake move towards food and consume it and search for next meal .

* 1. **SUMMARY PAGE**



**Fig 6.5: Summary page**

The Snake Game is a classic arcade style game where it is a single-player game but the focus is to achievethe highest score possible thus competing with yourself and others.

# CHAPTER 7

## CONCULSION

The player controls a long, thin creature, resembling a snake, which roams around on a bordered plane, picking up food 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.

## BIBLIOGRAPHY & REFERENCES

* OpenGL Programming Guide (Addison-Wesley Publishing Company)
* The OpenGL Utility Toolkit (GLUT)Programming Interface-API Version 3 BY MARK J. KILGARD
* Interactive computer graphics
* -A top down approach by using Open GL by EDWARD ANGEL
* Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version,3rd / 4th Edition, Pearson Education,2011
* Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th edition. Pearson Education, 2008
* James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer graphics with OpenGL: pearson education
* Xiang, Plastock : Computer Graphics , sham’s outline series, 2nd edition, TMG.

**List of websites***:*

* + 1. <http://www.opengl.org/>
    2. <http://www.academictutorials.com/graphics/graphics-flood-fill.asp>
    3. <http://www.glprogramming.com/>
    4. https://github.com/esrael-github/Computer-Graphics-Projects-/blob/main/Snake\_Game.rar

## APPENDIX

* + **GL:** Graphics Library
  + **GLU:** Graphics Utility Library
  + **GLUT:** OpenGL Utility Tool kit
  + **glutInitDisplayMode():** It sets the initial display mode.
  + **glutDisplayFunc():** sets the display callback for the *current window.*
  + **glutInitWindowPosition():** Initializes GLUT and specifies command-line options for window system in use.
  + **glColor():**Set the current color.
  + **2D:** Two dimentional.
  + **glLoadidentity():**This replaces the current matrix with the identity matrix.
  + **glutPostRedisplay():** This marks the current window as needing to be redisplayed.
  + **API:** Application Programming Interface.
  + **GLX:** OpenGL Extension to the X Window System.