Computer Graphics (UCS505)

Project on OPENGL based Snake Game

Submitted By

Jaskaran Singh Jaggi 101903103

Prakhar Bhateja 101903098

Ujjwal Verma 101903097

3COE-4

B.E. Third Year – COE

Submitted To:

Ms. Archana Kumari

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

Sr. No.	Description	Page No.
1.	Introduction to Project	3
2.	Computer Graphics concepts used	3
3.	User Defined Functions	4-5
4.	Source Code	6-15
5.	Output/ Screen shots	16-18

INTODUCTION TO PROJECT: OPENGL based Snake Game

Snake is a video game genre in which player controls a long, thin creature, resembling a snake, which roams around on a bordered plane. The concept originated in the 1976 Gremlin Industries two-player arcade game Blockade, which, due to its ease of implementation, has hundreds of versions (some titles include the word snake or worm) on many platforms.

The player controls a point, square, or object on a bounded plane. As it moves forward, it leaves a path that resembles a moving snake.

Basic functionalities of snake game:

• Snake can move in a given direction and when it eats the food, the length of snake increases.
• When snake crosses itself, or the wall, the game will over.
• Food will be generated at a given interval.

Computer Graphics concepts used :

- OpenGL (open graphics library) is a standard specification defining a cross-language across platform API for writing applications that produce 2D and 3D computer graphics.
- OpenGL is an application program interface (API) offering various functions to implement primitives, models, and images. It offers functions to create and manipulate render lighting, coloring, viewing the models, translation, rotation, and scaling of objects.
- The method of keyboard interaction was used heavily while making this game.

User Defined Functions:

initializeGame ()-

checks for correct input for various fields and initializes components required to start the game.

guide()-

After pressing key h the controls of the game are presented for easier understanding.

moveSnake()-

Moves the snake in a particular, specified direction. Initially the snake moves in the Right direction (using switch statement)

showFinalScore()-

Shows the final score of the game at the end of all lives.

Delay()-

Produces a delay of N*10^8 =N sec

Keyboard()-

Using switch, checks for w,s,a,d command for moving the snake, or p,h,m,r and q helper keys to produce required functions for further processing.

Display()-

Initialization of basic Opengl functions like glClear,glMatrixMode(),etc.

Game initialization functions like drawSnake(),drawFood(),drawWalls()

drawSnake()-

head_x value,head_y value ,making snake head, and body.Make a square eye part on the head.

drawFood()-

use rand() to create a random position inside the coordinate accessible to create food part.

(Not on parts of snake coordinates).

It also provides a special extra point food according to a lottery which also uses rand()

drawWalls()-

Creates the boundary of the game using 4 drawBrick() functions.

food_texture()-

draws a diamond shaped food at a randomly generated coordinate.

drawBrick()-

Used to generate square shaped brick at boundaries of frame.

TimerFunc()-

If not paused move snake in particular direction

SOURCE CODE:

```
#include <iostream>
#include <GL/glut.h>
#include <deque>
#include <assert.h>
using namespace std;

 // colors for the program in rgb format, use float values
#define mapBgColor   1.0,0.78,0.69
#define snakeColor   0.16,0.49,0.25
#define foodColor    0.95,0.44,0.125
#define wallColor    0.2,0.2,0.2
#define splFoodColor 1.0,0.0,0.0

int map_size = 20;      // keep value bw 10 and 50
int luck = 5;           // keep between 1 and 100 to get the
special food.
int initialLives = 5; // saves when you bit yourself
int maxDifficulty = 7; // keep atleast 5.

// assigning codes to directions
#define UP     1
#define DOWN   2
#define LEFT   3
#define RIGHT  4
#define delay(n) for(int i = 0; i < 1e5;i++)for(int j = 0; j <
n*1e3;j++);

deque<pair<int, int>> snake_body; // data structure for storing
snake cordinates
int food_pos[2];                  // x,y position of food.



int foodAvailable = 0;
int score = 0;
int special = 0;
int paused = 1;
int finished = 0;
int direction = RIGHT;
int Difficulty = maxDifficulty; //higher is easy
int currentLives = initialLives;
int gameSpeed = Difficulty * 20; // also timer val to be passed
in glutTimerFunc
```

```
void showFinalScore()
{
    cout << "**********************\n";
    cout << "Your Final Score is: " << score;
}

void guide()
{

printf("\n**************************************************
***");
    printf("\nUse WASD as movement keys.");
    printf("\nPress W for Up, A for left, S for Down and D for
Right");
    printf("\nM to change difficuly.");
    printf("\nR to Restart Game.");
    printf("\n+ to Increase map size.The game will restart
though.");
    printf("\n- to Decrease map size.The game will restart
though.");
    printf("\nP to pause/resume the game.");
    printf("\nPress Esc or Q to  Quit.");
    printf("\nPress H for help.");
    printf("\n\nPress P to Start.");

printf("\n**************************************************
***\n");
}

void draw_body_part(int x, int y)
{
    glColor3f(snakeColor);
    glBegin(GL_POLYGON); // a 1x1 square
    glVertex2i(x + 1, y + 1);
    glVertex2i(x + 1, y + 0);
    glVertex2i(x + 0, y + 0);
    glVertex2i(x + 0, y + 1);
    glEnd();

    glLineWidth(2);
    glColor3f(mapBgColor);
    glBegin(GL_LINES); // vertical line through middle of square
    glVertex2f(x + 0.5, y + 1);
    glVertex2f(x + 0.5, y + 0);
    glEnd();
    glBegin(GL_LINES); // horizontal line through middle of
square
```

```
    glVertex2f(x + 0, y + 0.5);
    glVertex2f(x + 1, y + 0.5);
    glEnd();
}

void food_texture(int x, int y)
{
    glBegin(GL_POLYGON);
    glVertex2f(x + 0.5, y + 1.0);
    glVertex2f(x + 0.0, y + 0.5);
    glVertex2f(x + 0.5, y + 0.0);
    glVertex2f(x + 1, y + 0.5);
    glEnd();
}

void drawBrick(int x, int y)
{
    glColor3f(wallColor);
    glBegin(GL_POLYGON);
    glVertex2f(x + 0.95, y + 0.95);
    glVertex2f(x + 0.05, y + 0.95);
    glVertex2f(x + 0.05, y + 0.05);
    glVertex2f(x + 0.95, y + 0.05);
    glEnd();

}

void drawWalls()
{
    for (int i = 0; i <= map_size;i++)
    {
        glColor3f(wallColor);
        drawBrick(i, 0);          // bottom wall
        drawBrick(i, map_size - 1); // top wall
        drawBrick(0, i);          // left wall
        drawBrick(map_size - 1, i); // right wall
    }
}

void drawFood()
{
    if (!foodAvailable)
    {
        // create food
        int fx = rand() % (map_size - 2) + 1, fy = rand() %
(map_size - 2) + 1;
        int lottery = 1 + rand() % 100;
```

```
        if (lottery <= luck && score != 0)
        {
            special = 1;
        }
        int overlap = 1;
        while (overlap) // make sure no overlap of food with
snake body
        {
            for (auto part : snake_body)
            {
                if (part.first == fx && part.second == fy)
                {
                    fx = rand() % map_size + 1, fy = rand() %
map_size + 1;
                    break;
                }
            }
            overlap = 0;
            food_pos[0] = fx;
            food_pos[1] = fy;
        }
    }
    foodAvailable = 1;

    if (special == 1)
        glColor3f(splFoodColor);
    else
        glColor3f(foodColor);

    food_texture(food_pos[0], food_pos[1]); // using a brick for
food
}

void drawSnake()
{
    glColor3f(snakeColor);
    int hx = snake_body[0].first;  // head x value
    int hy = snake_body[0].second; // head y value

    glBegin(GL_POLYGON); // 1x1 square
    glVertex2i(hx + 1, hy + 1);
    glVertex2i(hx + 1, hy + 0);
    glVertex2i(hx + 0, hy + 0);
    glVertex2i(hx + 0, hy + 1);
    glEnd();

    glColor3f(mapBgColor);
```

```
    glBegin(GL_POLYGON); // making a small square for eye of
snake
    glVertex2f(hx + 0.85, hy + 0.9);
    glVertex2f(hx + 0.85, hy + 0.65);
    glVertex2f(hx + 0.65, hy + 0.65);
    glVertex2f(hx + 0.65, hy + 0.9);
    glEnd();

    // draw body other than head
    glColor3f(snakeColor);
    int len = snake_body.size();
    for (int i = 1; i < len;i++)
    {
        draw_body_part(snake_body[i].first,
snake_body[i].second);
    }
}

void moveSnake(int newDirection)
{
    // A snake moves in the desired direction by adding new
blocks/elements
    // in front and popping block/element from the back so as to
give an illusion
    // that the snake is moving
    direction = newDirection;
    int delX = 0;
    int delY = 0;
    int mapEdge = 0;
    int snake_part_axis = 0;
    switch (direction)
    {
    case UP:
        delY = 1;
        break;
    case DOWN:
        delY = -1;
        break;
    case LEFT:
        delX = -1;
        break;
    case RIGHT:
        delX = 1;
        break;
    }

    // Check if snake gets bitten by itself or not
```

```
    for (auto part : snake_body)
    {
        if ((part.first == (snake_body[0].first + delX)) &&
(part.second == (snake_body[0].second + delY)))
        {
            currentLives--;
            cout << "Be Careful! You got bit.\n";
            if (currentLives <= 0)
            {
                cout << "Game Over.";
                showFinalScore();
                delay(2);
                exit(0);
            }
        }
    }

    // check if the snake runs into the wall
    if (snake_body[0].first <= 0 || snake_body[0].first >=
map_size - 1 || snake_body[0].second <= 0 ||
snake_body[0].second >= map_size - 1)
    {
        cout << "Oh NO! You ran into wall. Game Over.\n";
        showFinalScore();
        delay(2);
        exit(0);
    }

    // check if snake hits the food and make him grow by pushing
a new block inside
    //the deque without popping last entry
    int grow = 0;
    if (snake_body[0].first + delX == food_pos[0] &&
snake_body[0].second + delY == food_pos[1])
    {
        grow = 1;
        if (special)
        {
            grow = 100;
            special = 0;
            cout << "Legendary Food!!! +100 score.\n";
        }
        score += grow;
        foodAvailable = 0;
    }
```

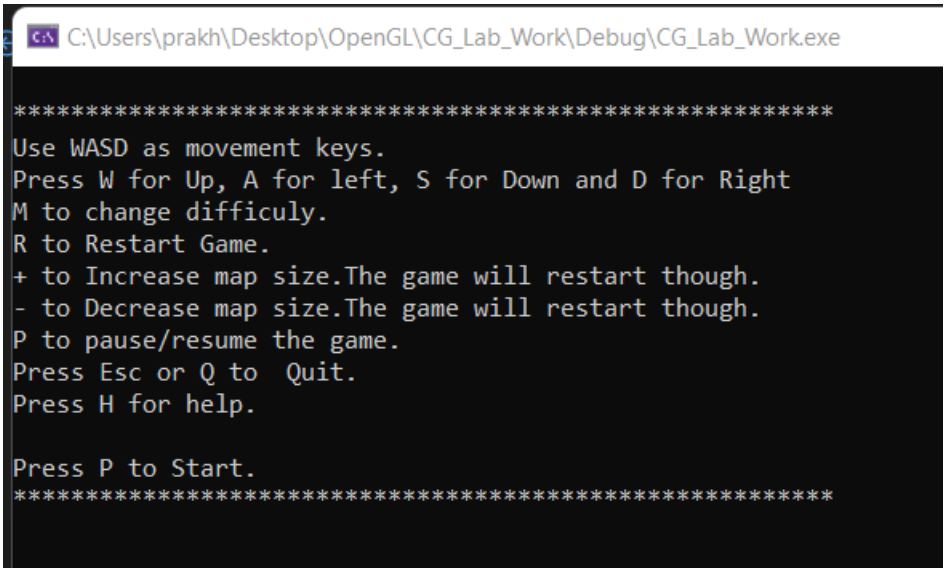
```
    // Whenever snake eats food we push a new block inside the
deque
    snake_body.push_front({ snake_body[0].first +
delX,snake_body[0].second + delY });

    // While the snake is moving we pop the blocks/elements from
the back of the
    // snakes body
    if (!grow)
    {
        snake_body.pop_back();
    }
    glutPostRedisplay();
}

void TimerFunc(int val)
{
    if (!paused)
        moveSnake(direction);
    glutTimerFunc(gameSpeed, TimerFunc, 0);
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0, map_size, 0, map_size);
    glMatrixMode(GL_MODELVIEW);
    drawSnake();
    drawFood();
    drawWalls();
    glutSwapBuffers();
}

void initializeGame()
{
    // some assertions
    assert(map_size >= 15);
    assert(map_size <= 50);
    assert(luck <= 100);
    assert(luck >= 1);
    assert(maxDifficulty >= 5);
    assert(initialLives > 0);
    srand(time(0));
```

```
    glClearColor(mapBgColor, 0);
    score = 0;
    currentLives = initialLives;

    // create size 3 snake
    snake_body.clear();
    snake_body.push_back({ 5,map_size / 2 });
    snake_body.push_back({ 4,map_size / 2 });
    snake_body.push_back({ 3,map_size / 2 });
    moveSnake(RIGHT);
    paused = 1;
    foodAvailable = 0;
    guide(); // show game guide
}

void keyboard(unsigned char key, int, int) {
    switch (key)
    {
        case 'W':
        case 'w':
        {
            if (!paused)
                if (direction == LEFT || direction == RIGHT) {
                    moveSnake(UP);
                }

            break;
        }
        case 'S':
        case 's':
        {
            if (!paused)
                if (direction == LEFT || direction == RIGHT) {
                    moveSnake(DOWN);
                }

            break;
        }
        case 'A':
        case 'a':
        {
            if (!paused)
                if (direction == UP || direction == DOWN) {
                    moveSnake(LEFT);
                }

            break;
```

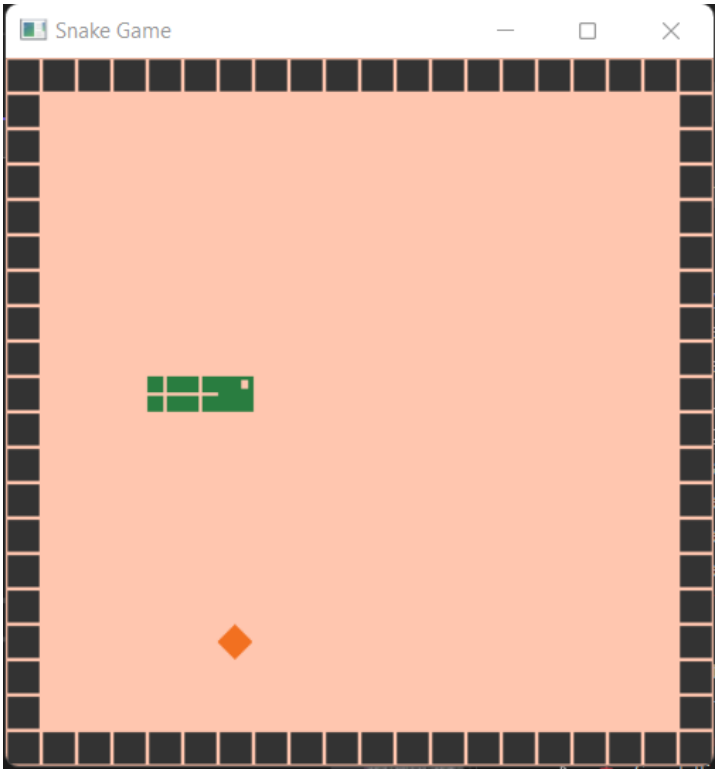
```
        }
        case 'D':
        case 'd':
        {
            if (!paused)
                if (direction == UP || direction == DOWN) {
                        moveSnake(RIGHT);
                }

            break;
        }
        case 'M':
        case 'm':
        {
            Difficulty = (Difficulty - 1) % (maxDifficulty + 1);
            if (Difficulty <= 0)
                Difficulty = maxDifficulty;
            gameSpeed = Difficulty * 20;
            cout << "New Difficulty is " << (maxDifficulty + 1)
- Difficulty << endl;
            break;
        }
        case 'P':
        case 'p':
        {   if (paused == 0)
            cout << "You Stopped\nPaused the game\n";
        else
            cout << "You Resumed/Started Game\n";
        paused = !paused;
        break;
        }
        case 'H':
        case 'h':
        {
            guide();
            break;
        }
        case 'R':
        case 'r':
        {

            showFinalScore();
            cout << "You restarted the
Game.\nRestarting...\n.All the best!!!.\n";
            initializeGame();
            break;
        }
```

```
        case 27:
        case 'q':
        {
            cout << "You pressed exit.\n";
            showFinalScore();
            exit(0);
        }
    }
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
    glutInitWindowSize(map_size * 20, map_size * 20);
    glutInitWindowPosition(250, 100);
    glutCreateWindow("Snake Game");
    glutDisplayFunc(display);
    glutKeyboardFunc(keyboard);
    glutTimerFunc(gameSpeed, TimerFunc, 0);
    initializeGame();
    glutMainLoop();
    return 0;
}
```

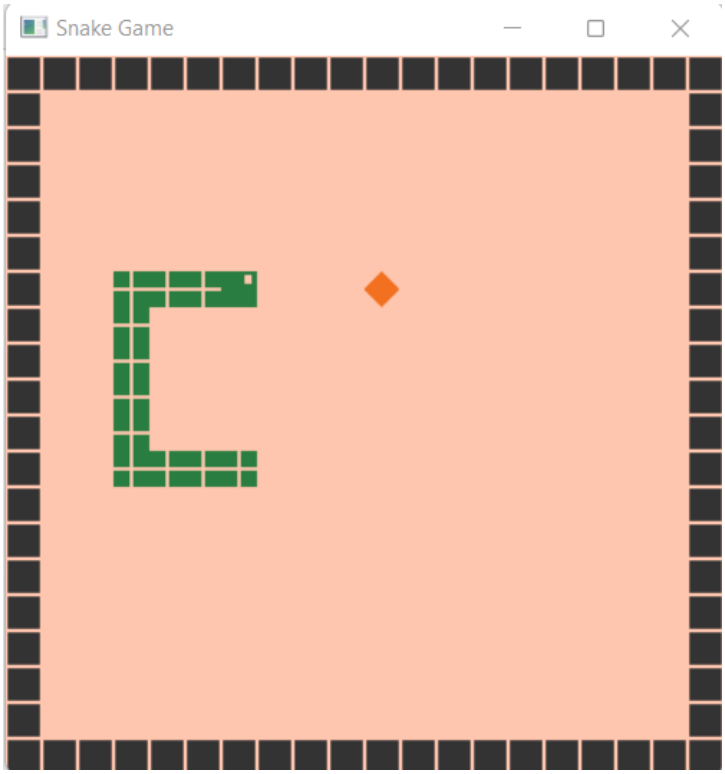
OUTPUT AND SCREENSHOTS:



Command Prompt used to display textual prompts like Instructions, Score etc.



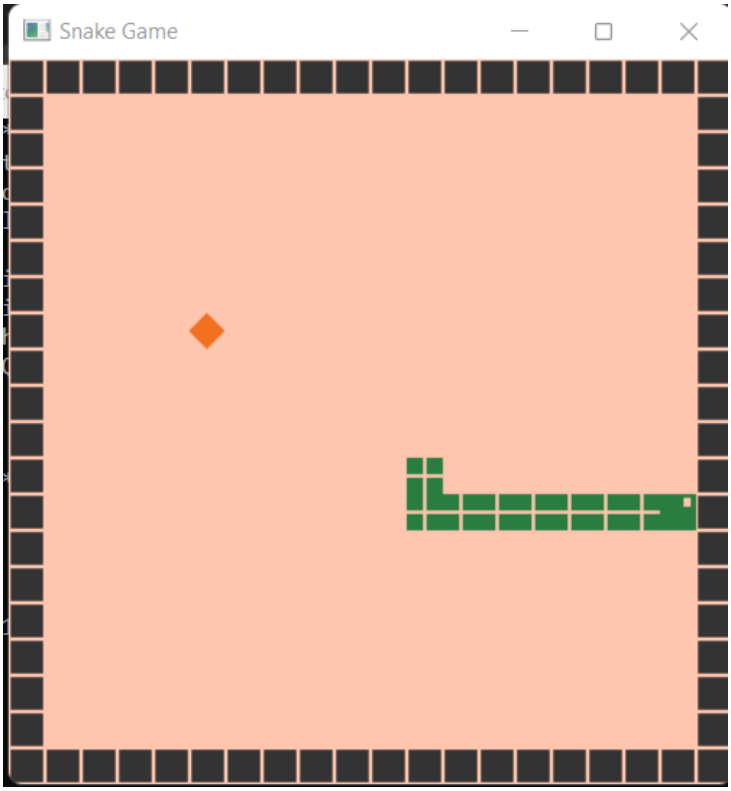
Initial position of the snake



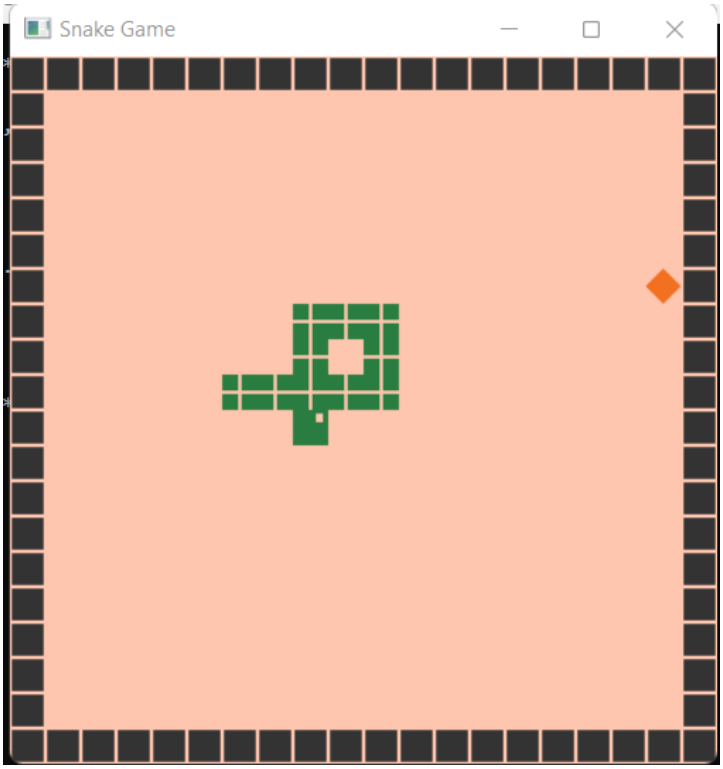
Snake gets bigger when it eats the orange food

```
**********************************************************
You Resumed/Started Game
Legendary Food!!! +100 score.
Legendary Food!!! +100 score.
Legendary Food!!! +100 score.
Be Careful! You got bit.
Legendary Food!!! +100 score.
Legendary Food!!! +100 score.
You Stopped
Paused the game
You Resumed/Started Game
Be Careful! You got bit.
Be Careful! You got bit.
Be Careful! You got bit.
Be Careful! You got bit.
Game Over.**********************
Your Final Score is: 509
C:\Users\prakh\Desktop\OpenGL\CG_Lab_Work\Debug\CG_Lab_Work.exe (process 10384) exited with code 0.
Press any key to close this window . . .
```

Command Prompt displays description of the events that are happening on the main game



Snake dies and game gets over whenever it hits the wall



Snake dies and game gets over whenever it crosses itself 5 times