Shivcharan T
205001100
CSE - B

SSN COLLEGE OF ENGINEERING, KALAVAKKAM
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Lab Exercise 7: Cohen Sutherland Line clipping in C++ using OpenGL

Apply Cohen Sutherland line clipping on a line (x1,y1) (x2,y2) with respect to a clipping window
(XWmin,YWmin) (XWmax,YWmax).
After clipping with respect to an edge, display the line segment with the calculated intermediate intersection points and the vertex list.
Input: The clipping window co-ordinates and the line endpoints
Note: The output should show the clipping window and the line to be clipped in different colors. You can show the intermediate steps using time delay.

Aim:
To implement Cohen Sutherland Line Clipping Algorithm in C++ using OpenGL

Algorithm:

```
1. Assign a region code for two endpoints of the given line.
2. If both endpoints have a region code 0000, then the given line is completely inside.
3. Else, perform the logical AND operation for both region codes.
  3.1. If the result is not 0000, then the given line is completely outside.
  3.2. Else, the line is partially inside.
     3.2.1. Choose an endpoint of the line that is outside the given rectangle.
     3.2.2. Find the intersection point of the rectangular boundary (based on the region code).
     3.2.3. Replace the endpoint with the intersection point and update the region code.
     3.2.4. Repeat step 2 until we find a clipped line either trivially accepted or trivially rejected.
4. Repeat step 1 for other lines.
```

Code:

5a.cpp:

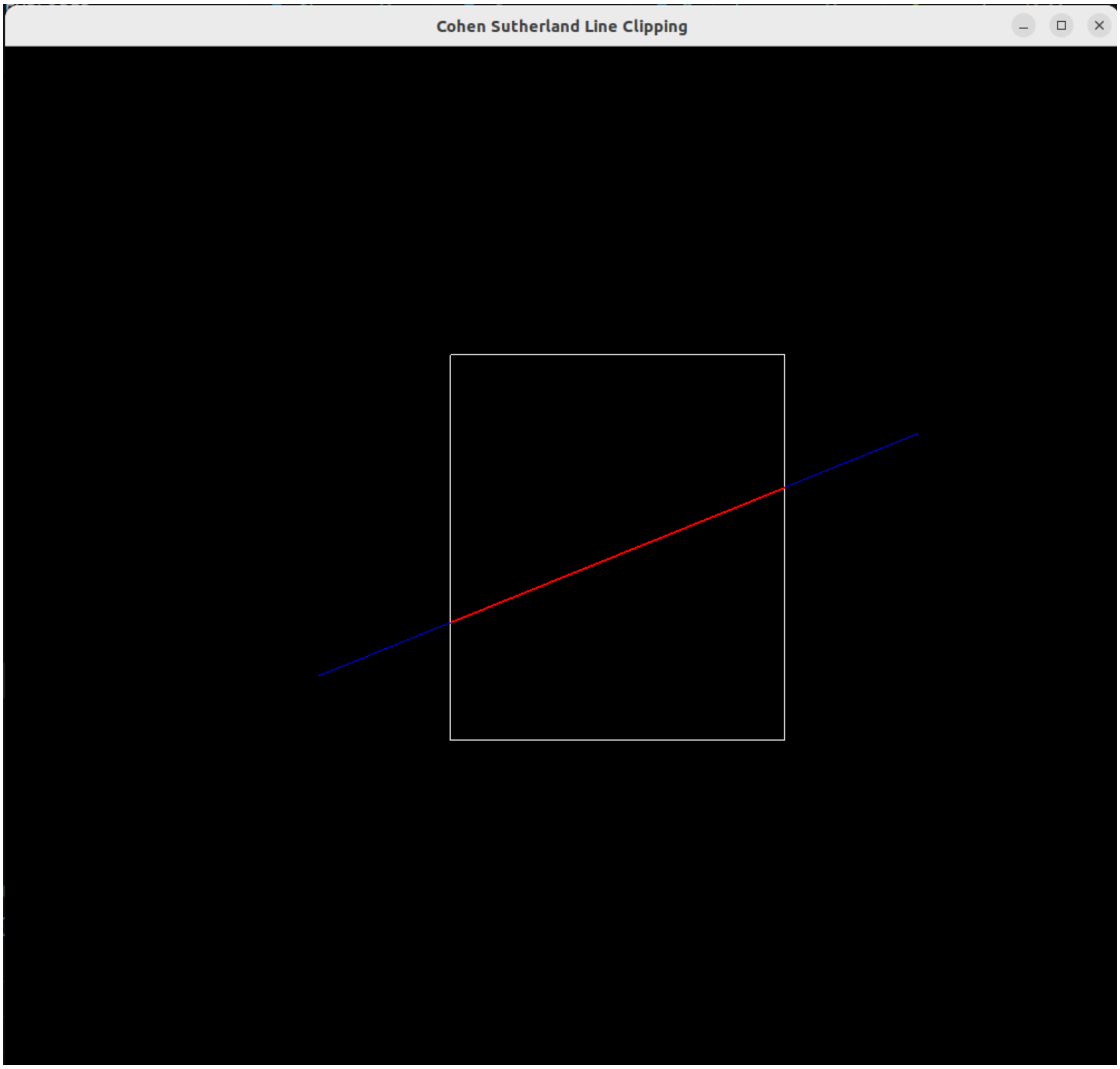
```
#include <cmath>
#include <cstring>
#include <stdio.h>
#include <iostream>
#include <GL/glut.h>
using namespace std;

int i = 0;

// mode
bool hardCode = false;
```

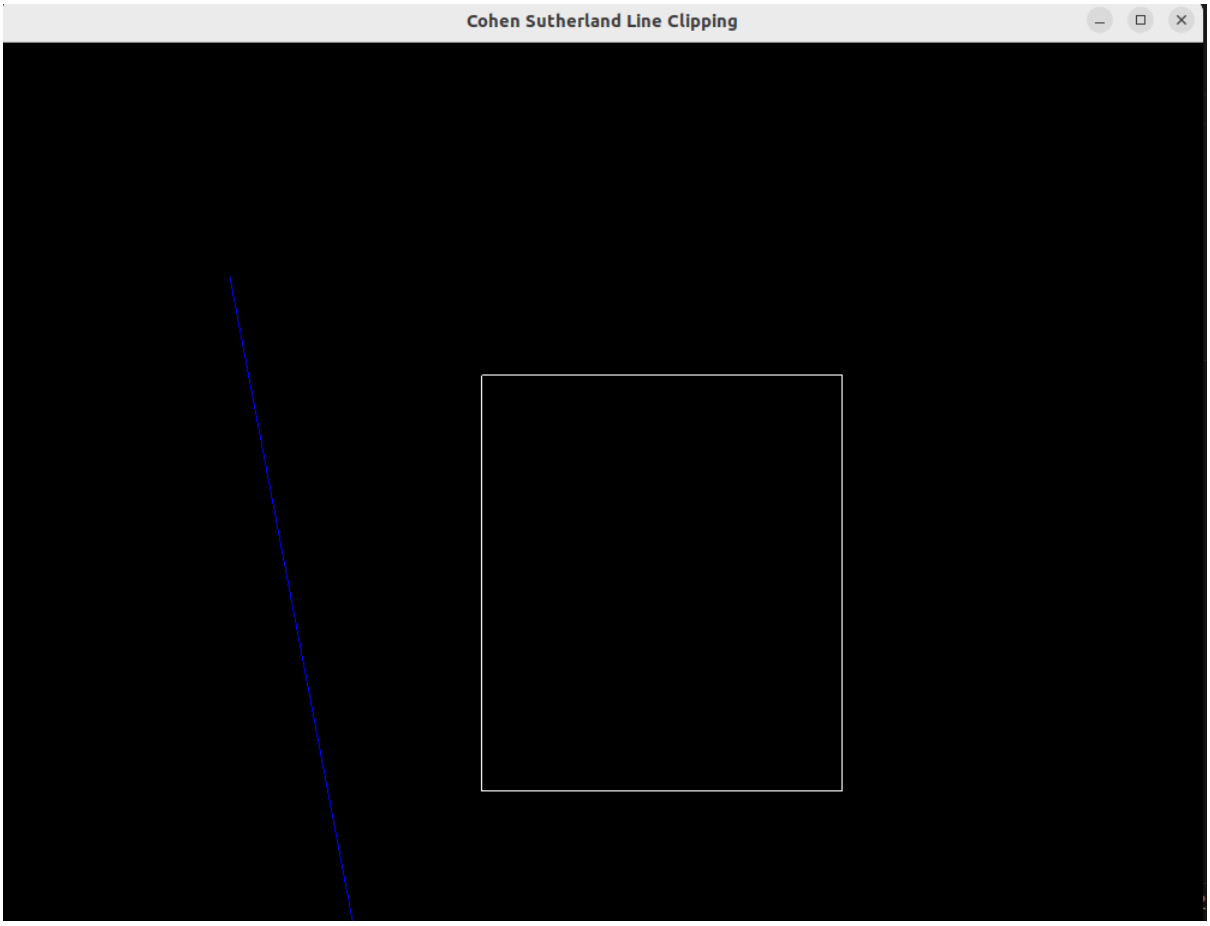
```
// screen dimensions
const int windowWidth = 1000;
const int windowHeight = 1000;

// TBRL
const int INSIDE = 0;
const int LEFT = 1;
const int RIGHT = 2;
const int BOTTOM = 4;
const int TOP = 8;

GLfloat xmin, xmax;
GLfloat ymin, ymax;

typedef struct
{
    GLfloat x, y;
} Point;

Point p1, p2;

void swap_points(Point *p1, Point *p2)
{
    Point t = *p1;
    *p1 = *p2;
    *p2 = t;
}

void swap_codes(GLint *x, GLint *y)
{
    GLint t = *x;
    *x = *y;
    *y = t;
}

GLint inside(GLint code)
{
    return !code;
}

GLint accept(GLint code1, GLint code2)
{
    return !(code1 | code2);
}

GLint reject(GLint code1, GLint code2)
{
    return code1 & code2;
}

int encode(Point p)
{
    GLfloat x = p.x, y = p.y;
    int code = INSIDE; // initialised as being inside of clip window
    if (y > ymax) // above the clip window
        code |= TOP;
    else if (y < ymin) // below the clip window
        code |= BOTTOM;
    if (x > xmax) // to the right of clip window
        code |= RIGHT;
    else if (x < xmin) // to the left of clip window
        code |= LEFT;
    return code; // return the calculated code
}
void myInit(void)
{
    glClearColor(0.0, 0.0, 0.0, 1.0);
    glColor3f(0.0, 1.0, 0.0);
    glPointSize(1.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-windowHeight / 2, windowHeight / 2, -windowWidth / 2, windowWidth
/ 2);
}

void draw_axis()
{
    glColor3f(1.0, 1.0, 1.0);
    glBegin(GL_LINE_LOOP);
    glVertex2d(-2000, 0);
    glVertex2d(2000, 0);

    glVertex2d(0, 2000);
    glVertex2d(0, -2000);
    glEnd();
    glFlush();
}

// GLint round(GLfloat a)
// {
//     return (GLint)(a + 0.5f);
// }

void line_clip(Point p1, Point p2)
{
    GLint code1, code2;
    GLint done = 0, plot_line = 0;
    GLfloat m = 0;
    if (p1.x != p2.x)
        m = (p2.y - p1.y) / (p2.x - p1.x);

    while (!done)
    {
        code1 = encode(p1);
        code2 = encode(p2);
        if (accept(code1, code2))
        {
            done = 1;
            plot_line = 1;
        }
        else if (reject(code1, code2))
        {
            done = 1;
        }
        else
        {
            if (inside(code1))
            {
                swap_points(&p1, &p2);
                swap_codes(&code1, &code2);
            }

            if (code1 & LEFT)
            {
                p1.y += (xmin - p1.x) * m;
                p1.x = xmin;
            }
            else if (code1 & RIGHT)
            {
                p1.y += (xmax - p1.x) * m;
                p1.x = xmax;
            }
            else if (code1 & BOTTOM)
            {
                if (p1.x != p2.x)
                    p1.x += (ymin - p1.y) / m;
                p1.y = ymin;
            }
            else if (code1 & TOP)
            {
                if (p1.x != p2.x)
                    p1.x += (ymax - p1.y) / m;
                p1.y = ymax;
            }
        }
    }

    if (plot_line)
    {
        glColor3f(1, 0, 0);
        glLineWidth(2);
        glBegin(GL_LINES);
        glVertex2i(round(p1.x), round(p1.y));
        glVertex2i(round(p2.x), round(p2.y));
        glEnd();
        glFlush();
    }
}

void draw_line()
{
    glBegin(GL_LINES);
    glVertex2i(round(p1.x), round(p1.y));
    glVertex2i(round(p2.x), round(p2.y));
    glEnd();
    glFlush();
}

void draw_window()
{
    glColor3f(1, 1, 1);
    glBegin(GL_LINE_LOOP);
    glVertex2i(round(xmin), round(ymin));
    glVertex2i(round(xmin), round(ymax));
    glVertex2i(round(xmax), round(ymax));
    glVertex2i(round(xmax), round(ymin));
    glEnd();
    glFlush();
}

void mykey(unsigned char ch, int x, int y)
{
    if (ch == 'c')
    {
        // clip_flag = 1;
        // glutPostRedisplay();

        line_clip(p1, p2);
        glFlush();
    }
}

void mymouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN && i < 2)
    {
        if (i == 0)
        {
            p1.x = x - windowWidth / 2;
            p1.y = windowHeight / 2 - y;
        }
        if (i == 1)
        {
            p2.x = x - windowWidth / 2;
            p2.y = windowHeight / 2 - y;
        }
        i++;
    }
    if (i == 2)
    {
        draw_line();
        i++;
    }
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    xmin = -100;
    xmax = +200;
    ymin = -123;
    ymax = +223;

    draw_window();

    glColor3f(0, 0, 1);

    if (hardCode)
    {
        p1 = {-450, -500};
        p2 = {250, 310};
        draw_line();
        // line_clip(p1, p2);   //default
    }

    // line_clip(p1, p2);   //default
    glFlush();
}
int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(windowHeight, windowWidth);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Cohen Sutherland Line Clipping");
    myInit();
    glutDisplayFunc(display);
    glutKeyboardFunc(mykey);
    glutMouseFunc(mymouse);
    glutMainLoop();
    return 1;
}
```

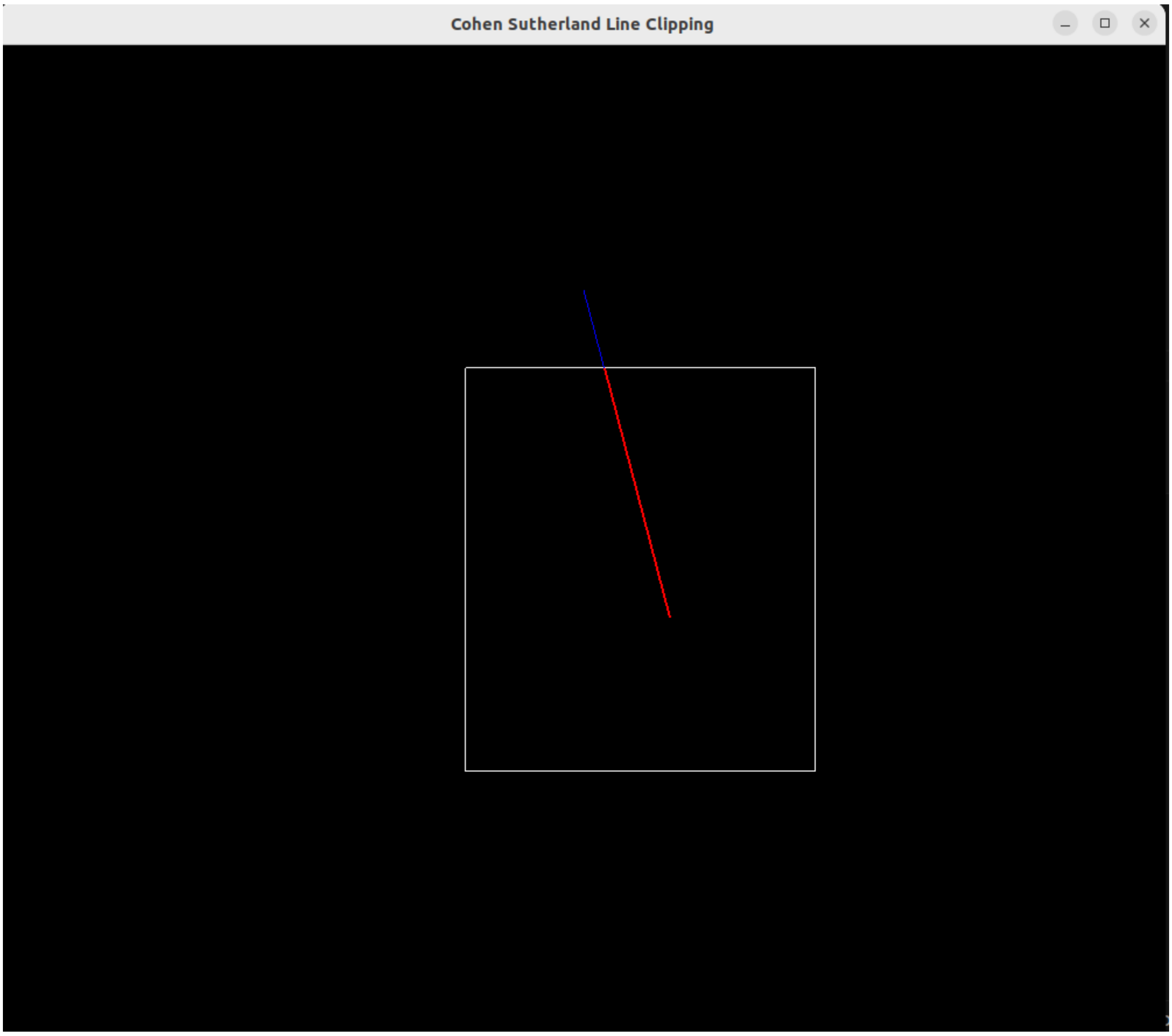
run.sh:

```
g++ 7.cpp -lGL -lglut -lGLU
./a.out
```

Sample I/O:


Cohen Sutherland Line Clipping


Cohen Sutherland Line Clipping



Learning Outcomes:

Learnt to do Line Clipping for a given window using Cohen Sutherland Line Clipping Algorithm.