****

**UCS1712-Graphics and Multimedia Lab**

**Programming Assignment 7**

**Cohen Sutherland Line clipping in C++ using OpenGL**

Name: Jayannthan P T

Dept: CSE ‘A’

Roll No.: 205001049

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 an edge, display the line segment with the calculated intermediate intersection points.

**Source code:**

#define GL\_SILENCE\_DEPRECATION

#include<GL/glut.h>

#include<iostream>

#include<math.h>

using namespace std;

float xmin, xmax, ymin, ymax;

char letter = 'A';

void myInit() {

    glClearColor(0.9, 0.9, 0.9, 0);

    glColor3f(0.0f, 0.0f, 0.0f);

    glPointSize(5);

    glMatrixMode(GL\_PROJECTION);

    glLineWidth(1.5);

    glLoadIdentity();

    gluOrtho2D(0.0, 640.0, 0.0, 480.0);

}

void labelPoint(float x, float y, char label) {

    glRasterPos2f(x + 320, y + 240);

    glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, label);

}

void displayPoint(float x, float y) {

    glBegin(GL\_POINTS);

    glVertex2d(x + 320, y + 240);

    glEnd();

}

void displayLine(int x1, int y1, int x2, int y2) {

    glBegin(GL\_LINES);

    glVertex2d(x1 + 320, y1 + 240);

    glVertex2d(x2 + 320, y2 + 240);

    glEnd();

}

void displayQuads(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {

    glBegin(GL\_QUADS);

    glVertex2d(x1 + 320, y1 + 240);

    glVertex2d(x2 + 320, y2 + 240);

    glVertex2d(x3 + 320, y3 + 240);

    glVertex2d(x4 + 320, y4 + 240);

    glEnd();

}

void drawPlane() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor4f(0.4, 0.4, 0.4, 1);

    displayLine(-320, 0, 320, 0);

    displayLine(0, -240, 0, 240);

    glFlush();

}

void printMatrix(float\* arr, int m, int n)

{

    int i, j;

    for (i = 0; i < m; i++) {

        for (j = 0; j < n; j++)

            cout << \*((arr + i \* n) + j) << " ";

        cout << endl;

    }

}

void printRegionCode(float\* RC) {

    printMatrix(RC, 4, 1);

}

float\* getRegionCode(float x, float y) {

    float\* TBRL = new float[4];

    TBRL[0] = (y > ymax) ? 1 : 0;

    TBRL[1] = (y < ymin) ? 1 : 0;

    TBRL[2] = (x > xmax) ? 1 : 0;

    TBRL[3] = (x < xmin) ? 1 : 0;

    return TBRL;

}

bool isTrivialAccept(float\* OR) {

    for (int i = 0; i < 4; i++) {

        if (OR[i] == 1) {

            cout << "not accept" << endl;

            return false;

        }

    }

    cout << "accept" << endl;

    return true;

}

bool isTrivialReject(float\* AND) {

    for (int i = 0; i < 4; i++) {

        if (AND[i] == 1) {

            cout << "reject" << endl;

            return true;

        }

    }

    cout << "not reject" << endl;

    return false;

}

int firstRegion(float\* OR) {

    int i;

    for (i = 0; OR[i] == 0; i++);

    return i;

}

void plotClippingWindow() {

    glColor4f(0.6, 0.6, 0.6, 1);

    displayQuads(xmin, ymin, xmin, ymax, xmax, ymax, xmax, ymin);

}

void plotInputLine(float x1, float y1, float x2, float y2) {

    glColor4f(0.0, 0.0, 0.5, 1);

    displayLine(x1, y1, x2, y2);

}

void applyCSLineClipping(float x1, float y1, float x2, float y2) {

    float\* RC1 = getRegionCode(x1, y1);

    cout << "\nRegion code of P1: " << endl;

    printRegionCode(RC1);

    cout << endl;

    float\* RC2 = getRegionCode(x2, y2);

    cout << "\nRegion code of P2: " << endl;

    printRegionCode(RC2);

    cout << endl;

    bool algoEnd = false;

    int green = 1;

    do {

        float\* OR = new float[4];

        float\* AND = new float[4];

        for (int i = 0; i < 4; i++) {

            OR[i] = max(RC1[i], RC2[i]);

            AND[i] = min(RC1[i], RC2[i]);

        }

        cout << "\nOR: " << endl;

        printRegionCode(OR);

        cout << "AND: " << endl;

        printRegionCode(AND);

        if (isTrivialAccept(OR) || isTrivialReject(AND)) {

            cout << "\nend" << endl;

            algoEnd = true;

            glColor4f(1, 0, 0, 1);

            displayLine(x1, y1, x2, y2);

        }

        else {

            int r = firstRegion(OR);

            cout << "r: " << r << endl;

            float x = x1, y = y1;

            float m = (y2 - y1) / (x2 - x1);

            glColor4f(0, green, 0, 1); green -= 0.2;

            if (r == 0) {

                y = ymax;

                x = x1 + (1 / m) \* (y - y1);

                RC1 = getRegionCode(x, y);

                x1 = x; y1 = y;

                labelPoint(x1, y1, letter++);

            }

            else if (r == 1) {

                y = ymin;

                x = x1 + (1 / m) \* (y - y1);

                RC2 = getRegionCode(x, y);

                x2 = x; y2 = y;

                labelPoint(x2, y2, letter++);

            }

            else if (r == 2) {

                x = xmax;

                y = y1 + m \* (x - x1);

                RC2 = getRegionCode(x, y);

                x2 = x; y2 = y;

                labelPoint(x2, y2, letter++);

            }

            else if (r == 3) {

                x = xmin;

                y = y1 + m \* (x - x1);

                RC1 = getRegionCode(x, y);

                x1 = x; y1 = y;

                labelPoint(x1, y1, letter++);

            }

            cout << "\nP1: " << x1 << " " << y1 << endl;

            cout << "P2: " << x2 << " " << y2 << endl;

            displayLine(x1, y1, x2, y2);

        }

    } while (!algoEnd);

}

void plotChart() {

    glClear(GL\_COLOR\_BUFFER\_BIT);

    drawPlane();

    cout << "COHEN SUTHERLAND LINE CLIPPING" << endl;

    cout << "\nEnter clipping window edges: " << endl;

    cout << "x\_min: "; cin >> xmin;

    cout << "x\_max: "; cin >> xmax;

    cout << "y\_min: "; cin >> ymin;

    cout << "y\_max: "; cin >> ymax;

    plotClippingWindow();

    float x1, y1, x2, y2;

    cout << "\nEnter line endpoints: " << endl;

    cout << "P1: "; cin >> x1 >> y1;

    cout << "P2: "; cin >> x2 >> y2;

    if (x2 < x1) {

        cout << "Swapping points so that P1 lies to the left of P2..." << endl;

        float tx = x1, ty = y1;

        x1 = x2; y1 = y2;

        x2 = tx; y2 = ty;

    }

    plotInputLine(x1, y1, x2, y2);

    labelPoint(x1, y1, letter++);

    labelPoint(x2, y2, letter++);

    applyCSLineClipping(x1, y1, x2, y2);

    glFlush();

}

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

    glutInit(&argc, argv);

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGBA);

    glutInitWindowSize(640, 480);

    glutCreateWindow("Cohen Sutherland Line Clipping");

    glutDisplayFunc(plotChart);

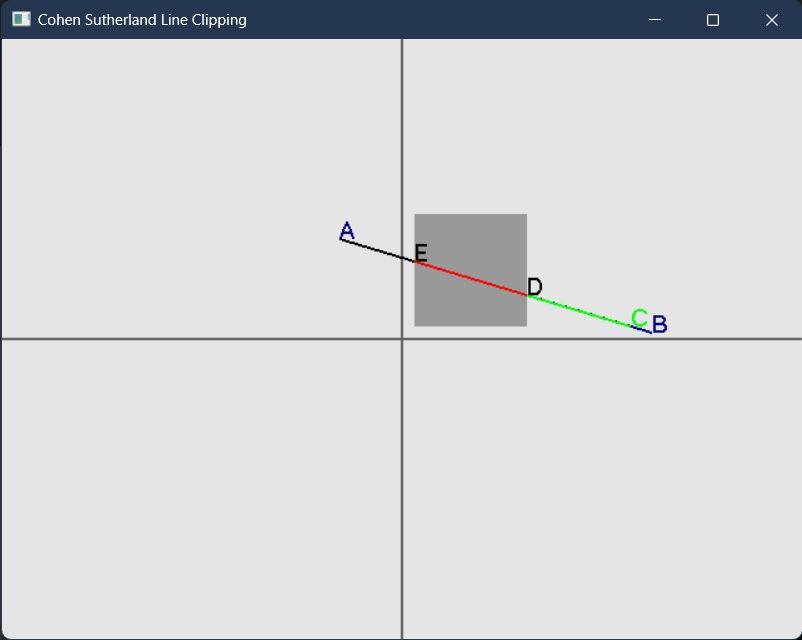
    myInit();

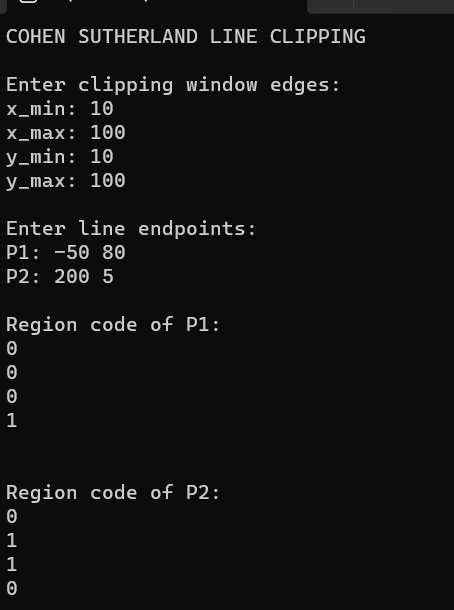
    glutMainLoop();

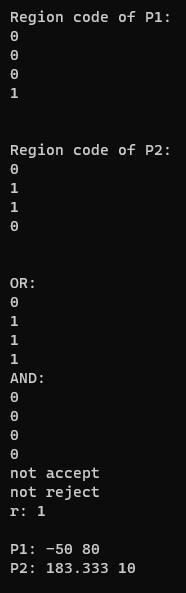
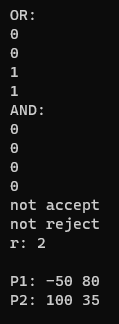
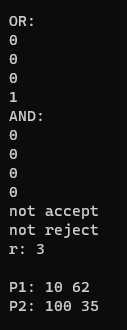
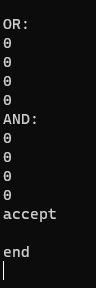
    return 1;

}

**Output**

****

****

**   **