UCS 1712 – GRAPHICS AND MULTIMEDIA LAB

ASSIGNMENT – 7

VISHAL N

185001198

CSEC

26.09.2021

1. COHEN_SUTHERLAND.CPP:

```
#include <GL/glut.h>
#include <cmath>
#include <iostream>

using namespace std;

typedef struct point {
    double x;
    double y;
};
typedef struct line {
    point A;
    point B;
    double m;
};
typedef struct window {
    int xmin;
    int xmax;
    int ymin;
    int ymax;
};
void initialize() {
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glColor3f(0.0f, 0.0f, 0.0f);
    glPointSize(2);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-250.0, 250.0, -250.0, 250.0);
}
void getRegionCode(point X, window W, char rc[4]) {
    for (int i = 0; i < 4; i++)
        rc[i] = '0';
    if (X.x < W.xmin) {
        rc[3] = '1';
    }
```

```
    if (X.x > W.xmax) {
        rc[2] = '1';
    }
    if (X.y < W.ymin) {
        rc[1] = '1';
    }
    if (X.y > W.ymax) {
        rc[0] = '1';
    }
}
point getIntersectionPoint(point X, double m, window W, char d) {
    point P;
    P.x = 0;
    P.y = 0;
    switch (d) {
    case 'T':
        P.y = W.ymax;
        P.x = X.x + (1 / m) * (P.y - X.y);
        break;
    case 'B':
        P.y = W.ymin;
        P.x = X.x + (1 / m) * (P.y - X.y);
        break;
    case 'R':
        P.x = W.xmax;
        P.y = X.y + m * (P.x - X.x);
        break;
    case 'L':
        P.x = W.xmin;
        P.y = X.y + m * (P.x - X.x);
        break;
    }
    return P;
}
point getClippedPoint(point X, double m, window W) {
    point P;
    P.x = X.x;
    P.y = X.y;
    char regions[5] = "TBRL";
    char rc[5];
    rc[4] = '\0';
    getRegionCode(X, W, rc);
    for (int i = 0; i < 4; i++) {
        if (rc[i] == '1') {
            P = getIntersectionPoint(X, m, W, regions[i]);
            rc[i] = '0';
            if (P.x >= W.xmin && P.x <= W.xmax && P.y >= W.ymin && P.y <=
W.xmax)
```

```
                break;
            }
        }
        return P;
}
line clip(line L, window W) {
        line clipped;
        clipped.A = getClippedPoint(L.A, L.m, W);
        clipped.B = getClippedPoint(L.B, L.m, W);
        return clipped;
}
void clippingDemo() {
        glClear(GL_COLOR_BUFFER_BIT);
        line L, C;
        window W;
        cout << "-----INPUT-----" << "\n";
        cout << "Enter xMin, xMax, yMin, yMax: ";
        cin >> W.xmin >> W.xmax >> W.ymin >> W.ymax;
        cout << "Enter line endpoint 1: ";
        cin >> L.A.x >> L.A.y;
        cout << "Enter line endpoint 2: ";
        cin >> L.B.x >> L.B.y;
        L.m = (L.A.y - L.B.y) / (L.A.x - L.B.x);
        C = clip(L, W);
        glBegin(GL_LINES);
        glVertex2d(-250, 0);
        glVertex2d(250, 0);
        glVertex2d(0, 250);
        glVertex2d(0, -250);
        glEnd();
        glBegin(GL_LINES);
        glVertex2f(W.xmin, W.ymin);
        glVertex2f(W.xmin, W.ymax);
        glVertex2f(W.xmin, W.ymax);
        glVertex2f(W.xmax, W.ymax);
        glVertex2f(W.xmax, W.ymax);
        glVertex2f(W.xmax, W.ymin);
        glVertex2f(W.xmax, W.ymin);
        glVertex2f(W.xmin, W.ymin);
        glEnd();
        glBegin(GL_LINES);
        glColor3f(1.0f, 0.0f, 0.0f);
        glVertex2f(L.A.x, L.A.y);
        glVertex2f(L.B.x, L.B.y);
        glEnd();
        glBegin(GL_LINES);
        glColor3f(0.0f, 0.0f, 1.0f);
        glVertex2f(C.A.x, C.A.y);
```

```
    glVertex2f(C.B.x, C.B.y);
    glEnd();
    glFlush();
}
int main(int argc, char* argv[]) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(500, 500);
    glutCreateWindow("Cohen Sutherland Line Clipping");
    glutDisplayFunc(clippingDemo);
    initialize();
    glutMainLoop();
}
```

OUTPUT