

SSN COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS1712 – GRAPHICS AND MULTIMEDIA LAB

Name : Rahul Ram M

Reg .No : 185001121

Date : 15/09/2021

EX NO: 7

Cohen Sutherland Line Clipping Algorithm

Aim:

To write a C++ program using OPENGGL to clip a line using the Cohen – Sutherland line clipping algorithm.

Algorithm:

1. Defining the bit sequence for TOP, BOTTOM, RIGHT, LEFT and INSIDE which will be used to identify whether the points lie top, bottom, right, left and inside respectively wrt clipping boundaries.
2. Creating a function to compute the regional code of both the vertices.
3. Clipping Vertical boundaries:
 - a. Find the slope of the line $m = (y_2 - y_1) / (x_2 - x_1)$.
 - b. If any point lie to the left of the vertical boundary,
 - i. Make x as x_{\min} (min x value of the boundary).
 - ii. Find y as $y = y_1 + m(x - x_1)$
 - c. If any point lie to the right of the vertical boundary,
 - i. Make x as x_{\max} (max x value of the boundary).
 - ii. Find y as $y = y_1 + m(x - x_1)$
 - d. Create a window named Clipping vertical boundaries and draw the boundaries and the lines.
4. Clipping Vertical boundaries:

- a. Find the slope of the line $m = (y_2 - y_1) / (x_2 - x_1)$.
- b. If any point lie to the bottom the horizontal boundary,
 - i. Make y as y_{\min} (min y value of the boundary).
 - ii. Find x as $x = x_1 + (y - y_1) / m$
- c. If any point lie to the top of the horizontal boundary,
 - i. Make y as y_{\max} (max y value of the boundary).
 - ii. Find x as $x = x_1 + (y - y_1) / m$
- d. Create a window named Clipping horizontal boundaries and draw the boundaries and the lines.

Code:

```
#include <GL/glut.h>
#include <math.h>
#include <iostream>
#include <vector>

using namespace std;

float X1[3], Y1[3], X2[3], Y2[3];

float x_min, y_min, x_max, y_max;

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

void myInit()
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    glPointSize(1);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0, 800, 0, 600);
}

int computeCode(float x, float y)
{
    int code = INSIDE;
```

```

    if (x < x_min)
        code |= LEFT;
    else if (x > x_max)
        code |= RIGHT;
    if (y < y_min)
        code |= BOTTOM;
    else if (y > y_max)
        code |= TOP;

    return code;
}

void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINE_LOOP);
    glVertex2f(x_min, y_min);
    glVertex2f(x_max, y_min);
    glVertex2f(x_max, y_max);
    glVertex2f(x_min, y_max);
    glEnd();

    glBegin(GL_LINES);
    glVertex2f(X1[0], Y1[0]);
    glVertex2f(X2[0], Y2[0]);
    glEnd();

    glFlush();
}

void displayX(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINE_LOOP);
    glVertex2f(x_min, y_min);
    glVertex2f(x_max, y_min);
    glVertex2f(x_max, y_max);
    glVertex2f(x_min, y_max);
    glEnd();
}

```

```

    glBegin(GL_LINES);
    glVertex2f(X1[1], Y1[1]);
    glVertex2f(X2[1], Y2[1]);
    glEnd();

    glFlush();
}

void displayY(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glBegin(GL_LINE_LOOP);
    glVertex2f(x_min, y_min);
    glVertex2f(x_max, y_min);
    glVertex2f(x_max, y_max);
    glVertex2f(x_min, y_max);
    glEnd();

    glBegin(GL_LINES);
    glVertex2f(X1[2], Y1[2]);
    glVertex2f(X2[2], Y2[2]);
    glEnd();

    glFlush();
}

void clipX()
{
    int codeP1 = computeCode(X1[0], Y1[0]);
    int codeP2 = computeCode(X2[0], Y2[0]);

    float m = (Y2[0] - Y1[0]) / (X2[0] - X1[0]);

    X1[1] = X1[0]; Y1[1] = Y1[0];
    X2[1] = X2[0]; Y2[1] = Y2[0];

    if (codeP1 & LEFT)
    {
        Y1[1] = Y1[0] + (m * (x_min - X1[0]));
        X1[1] = x_min;
    }
    if (codeP2 & LEFT)

```

```

{
    Y2[1] = Y1[0] + (m * (x_min - X1[0]));
    X2[1] = x_min;
}

if (codeP1 & RIGHT)
{
    Y1[1] = Y1[0] + (m * (x_max - X1[0]));
    X1[1] = x_max;
}
if (codeP2 & RIGHT)
{
    Y2[1] = Y1[0] + (m * (x_max - X1[0]));
    X2[1] = x_max;
}
}

void clipY()
{
    int codeP1 = computeCode(X1[1], Y1[1]);
    int codeP2 = computeCode(X2[1], Y2[1]);

    float m = (Y2[0] - Y1[0]) / (X2[0] - X1[0]);

    X1[2] = X1[1]; Y1[2] = Y1[1];
    X2[2] = X2[1]; Y2[2] = Y2[1];

    if (codeP1 & BOTTOM)
    {
        X1[2] = X1[1] + ((y_min - Y1[1]) / m);
        Y1[2] = y_min;
    }
    if (codeP2 & BOTTOM)
    {
        X2[2] = X1[1] + ((y_min - Y1[1]) / m);
        Y2[2] = y_min;
    }

    if (codeP1 & TOP)
    {
        X1[2] = X1[1] + ((y_max - Y1[1]) / m);
        Y1[2] = y_max;
    }
}

```

```

    if (codeP2 & TOP)
    {
        X2[2] = X1[1] + ((y_max - Y1[1]) / m);
        Y2[2] = y_max;
    }
}

int main(int argc, char** argv)
{
    cout << "Window Boundaries:\n";
    cout << "X min : "; cin >> x_min;
    cout << "Y min : "; cin >> y_min;
    cout << "X max : "; cin >> x_max;
    cout << "Y max : "; cin >> y_max;

    cout << "Point 1:\n";
    cout << "X : "; cin >> X1[0];
    cout << "Y : "; cin >> Y1[0];
    cout << "Point 2:\n";
    cout << "X : "; cin >> X2[0];
    cout << "Y : "; cin >> Y2[0];

    glutInit(&argc, argv); // Initialize

    glutInitWindowSize(800, 600); // Set the window's initial width &
height
    glutInitWindowPosition(50, 50);
    glutCreateWindow("Before Clipping");
    myInit();
    glutDisplayFunc(display);

    glutInitWindowSize(800, 600); // Set the window's initial width &
height
    glutInitWindowPosition(150, 150);
    glutCreateWindow("Clipping Vertical Boundaries");
    myInit();
    clipX();
    glutDisplayFunc(displayX);

    glutInitWindowSize(800, 600); // Set the window's initial width &

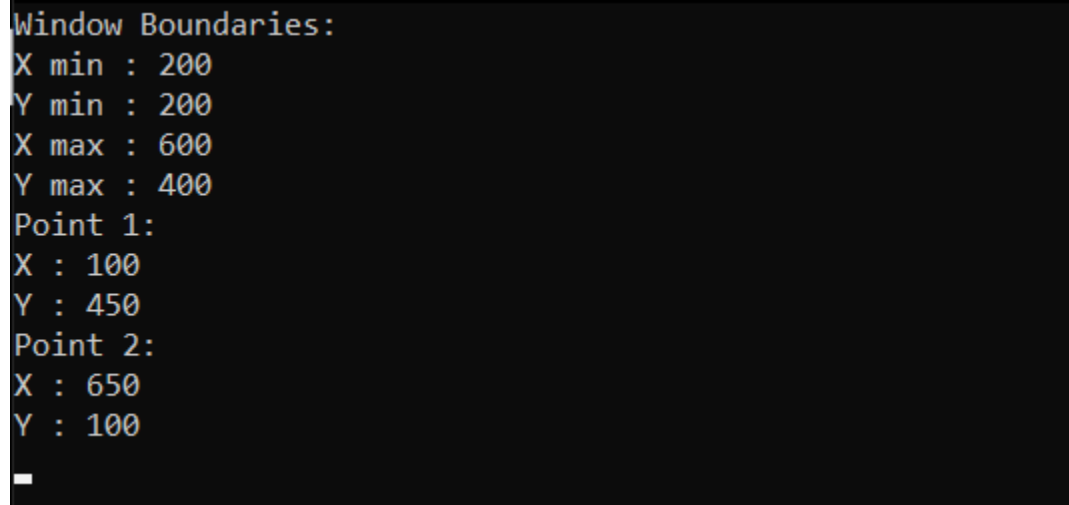
```

```
height
    glutInitWindowPosition(250, 250);
    glutCreateWindow("Clipping Horizontal Boundaries");
    myInit();
    clipY();
    glutDisplayFunc(displayY);

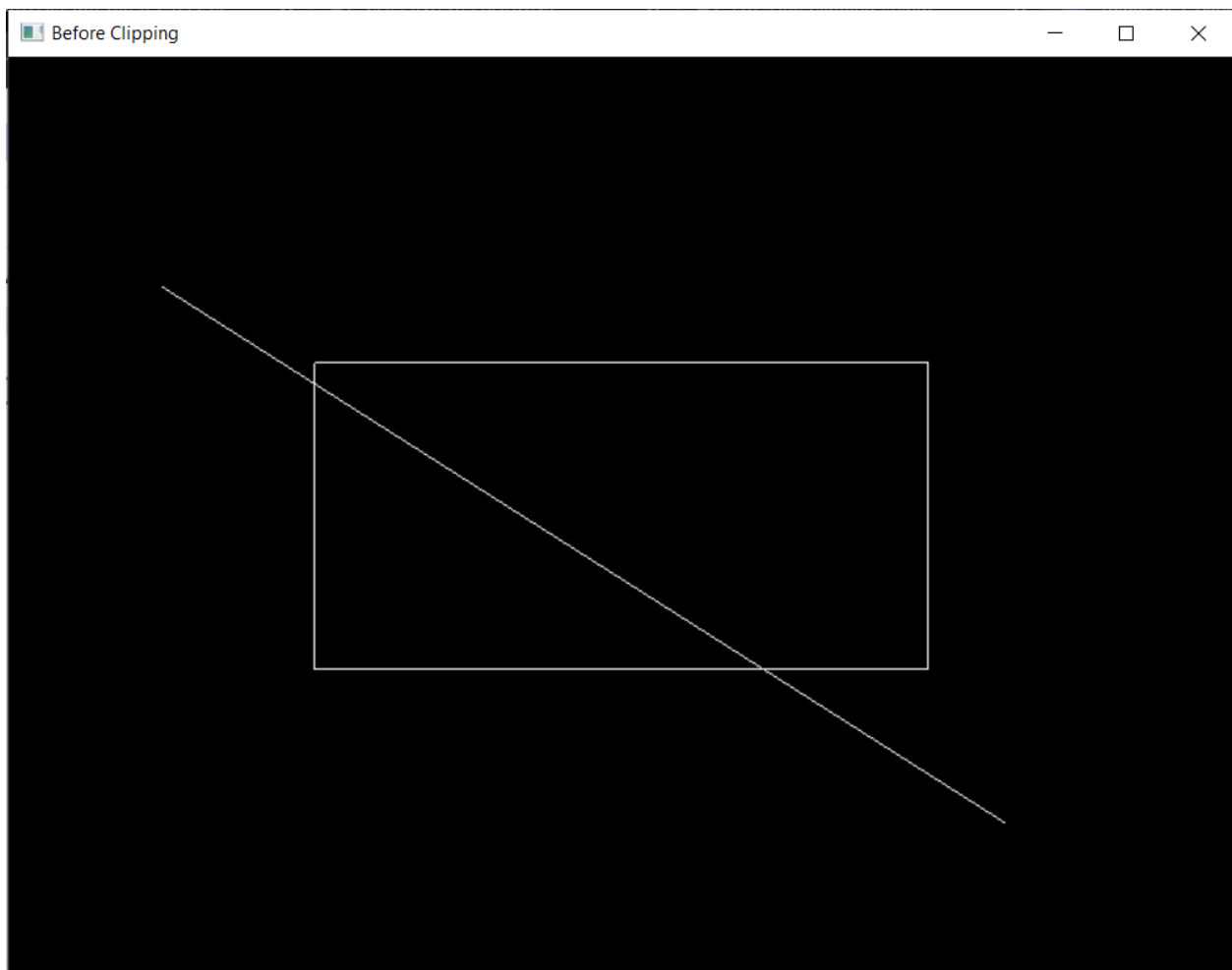
    glutMainLoop();
    return 0;
}

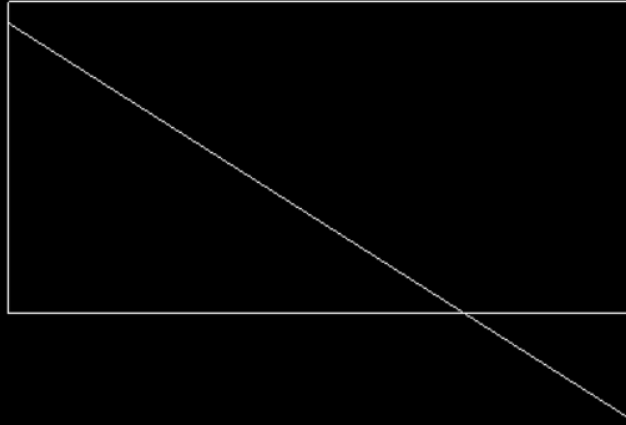
// 200 200 600 400 100 450 650 100
```

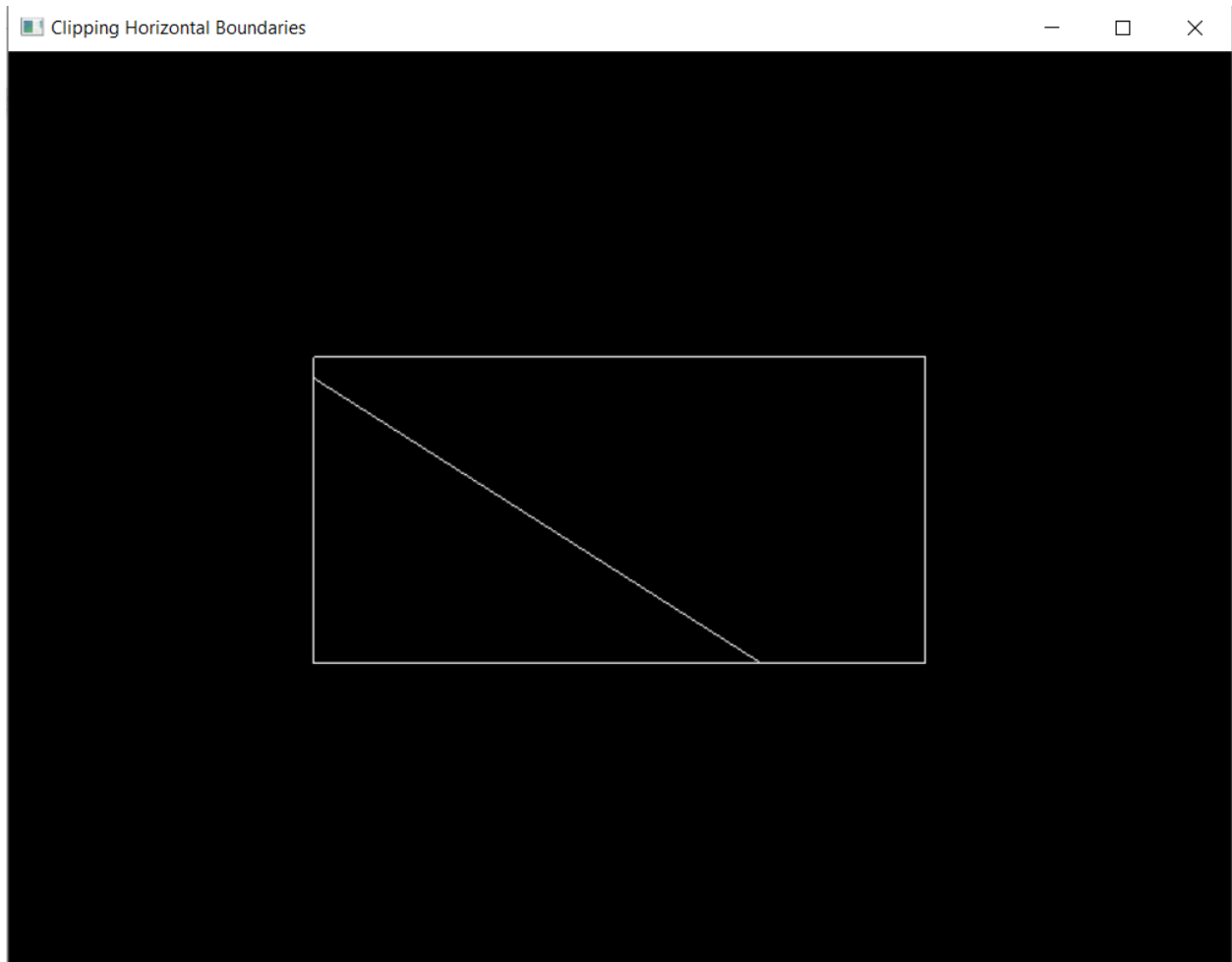
Input/Output Screenshots:



```
Window Boundaries:
X min : 200
Y min : 200
X max : 600
Y max : 400
Point 1:
X : 100
Y : 450
Point 2:
X : 650
Y : 100
_
```







Result:

Hence a C++ program is written using OPENGL to clip a line using Cohen – Sutherland line clipping algorithm and three different windows are created to show the clipping as before clipping, after clipping vertical boundaries and after clipping horizontal boundaries.