SSN COLLEGE OF ENGINEERING, KALAVAKKAM DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

UCS1712 - GRAPHICS AND MULTIMEDIA LAB

Assignment- 7 - Cohen Sutherland Line Clipping Algorithm

Name: Sabarivasan V **Reg.No: 205001085**

Aim:

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.

ALGORITHM:

1) Assign the region codes to both endpoints. 2)
Perform OR operation on both of these
endpoints. 3) if OR = 0000, then it is completely visible
(inside the window).
Else
Perform AND operation on both these endpoints.
i) if AND ≠ 0000,
then the line is invisible and not inside the window. Also, it can't be considered for clipping. ii) else
AND = 0000, the line is partially inside the window and considered for
Clipping.

4) After confirming that the line is partially inside the window, then we find the intersection with the boundary of the window. By using the following formula:-

Slope:- m= (y2-y1)/(x2-x1)

a) If the line passes through top or the line intersects with the top boundary of the window.
x = x + (y_wmax – y)/m y = y_wmax

b) If the line passes through the bottom or the line intersects with the bottom boundary of the window. x = x +
(y_wmin – y)/m y =
y_wmin

c) If the line passes through the left region or the line intersects with the left boundary of the window.

y = y+ (x_wmin – x)*m
x = x_wmin

d) If the line passes through the right region or the line intersects with the right boundary

of the window.

y = y + (x_wmax

-x)*m x = x_wmax

5) Now, overwrite the endpoints with a new one and update it.

6) Repeat the 4th step till your line doesn't get completely clipped

CODE:

```
#include <iostream>
#include<GLUT/glut.h>
using namespace std;
void myInit() {
glClearColor(1,1,1,0.0);
glColor3f(1.0f,1.0f,1.0f);
glPointSize(3);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,640.0,0.0,480.0);
}
// Defining region codes
const int INSIDE = 0; //
0000 const int LEFT = 1; //
0001 const int RIGHT = 2; //
0010 const int BOTTOM = 4; //
0100 const int TOP = 8; // 1000
// Defining x_max, y_max and x_min, y_min for
// clipping rectangle. Since diagonal points are
// enough to define a rectangle
const int x_max = 400;
const int y_max
= 400; const int x_min
= 200; const int y_min
= 200;
// Function to compute region code for a point(x, y)
int computeCode(double x, double y)
{
// initialized as being inside int code =
INSIDE; if (x < x_min) // to the left of
rectangle code |= LEFT; else if (x > x_max)
// to the right of rectangle code |= RIGHT;
if (y < y_min) // below the rectangle
code |= BOTTOM;
```

```
else if (y > y_max) // above the rectangle
code |= TOP; return code;
}
// Implementing Cohen-Sutherland algorithm //
Clipping a line from P1 = (x2, y2) to P2 = (x2, y2)
void lineclip(double x1, double y1, double x2,
double y2)
{
// Compute region codes for P1, P2
int code1 = computeCode(x1, y1);
int code2 = computeCode(x2, y2);
// Initialize line as outside the rectangular window
bool accept = false; while (true) { if ((code1 ==
0) && (code2 == 0)) { // If both endpoints lie
within rectangle accept = true; break; } else if
(code1 & code2) {
// If both endpoints are outside rectangle,
// in same region
break; } else {
// Some segment of line lies within the
// rectangle int
code_out;
double x=0, y=0;
// At least one endpoint is outside the
// rectangle, pick it. if (code1 != 0)
code_out = code1; else
code_out = code2;
// Find intersection point;
// using formulas y = y1 + slope * (x - x1),
// x = x1 + (1 / slope) * (y - y1) if
(code_out & TOP) {
// point is above the clip rectangle x = x1 +
(x2 - x1) * (y_max - y1) / (y2 - y1); y =
y_max; } else if (code_out & BOTTOM) {
// point is below the rectangle x = x1 + (x2 -
x1) * (y_min - y1) / (y2 - y1); y = y_min; }
else if (code_out & RIGHT) { // point is to
the right of rectangle y = y1 + (y2 - y1) *
(x_max - x1) / (x2 - x1); x = x_max; } else
if (code_out & LEFT) { // point is to the left
of rectangle y = y1 + (y2 - y1) * (x_min -
x1) / (x2 - x1); x = x_min;
}
```

```
// Now intersection point x, y is found
// We replace point outside rectangle
// by intersection point if (code_out
== code1) { x1 = x; y1 = y; code1 =
computeCode(x1, y1);
} else { x2 = x; y2 = y; code2
= computeCode(x2, y2);
}
} } if (accept) { cout << "Line accepted from
" << x1 << ", " << y1 << " to " << x2 << ", "
<< y2 << endl; glBegin(GL_LINES);
glColor3f(0.5f,0.1f,0.5f);
glVertex2d(x1,y1);
glVertex2d(x2,y2);
glEnd(); glFlush(); }
else cout << "Line rejected" <<
endl; } // Driver code void
myDisplay() { cout<<"The
clipping window has been set
with the coordinates:\n";
cout<<"(200,200),(200,400),(40
0,400) and (400,200)\n\n";
// First Line segment
int x1,x2,y1,y2;
cout<<"Enter (x1,y1):";
cin>>x1>>y1;
cout<<"Enter (x2,y2):";
cin>>x2>>y2;
glFlush();
glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_LINES);
glColor3f(0.5f,0.1f,0.1f);
glVertex2d(200,200);
glVertex2d(200,400);
glEnd(); glFlush();
glBegin(GL_LINES);
glColor3f(0.5f,0.1f,0.1f);
glVertex2d(200,400);
glVertex2d(400,400);
glEnd(); glFlush();
glBegin(GL_LINES);
glColor3f(0.5f,0.1f,0.1f);
glVertex2d(400,400);
```

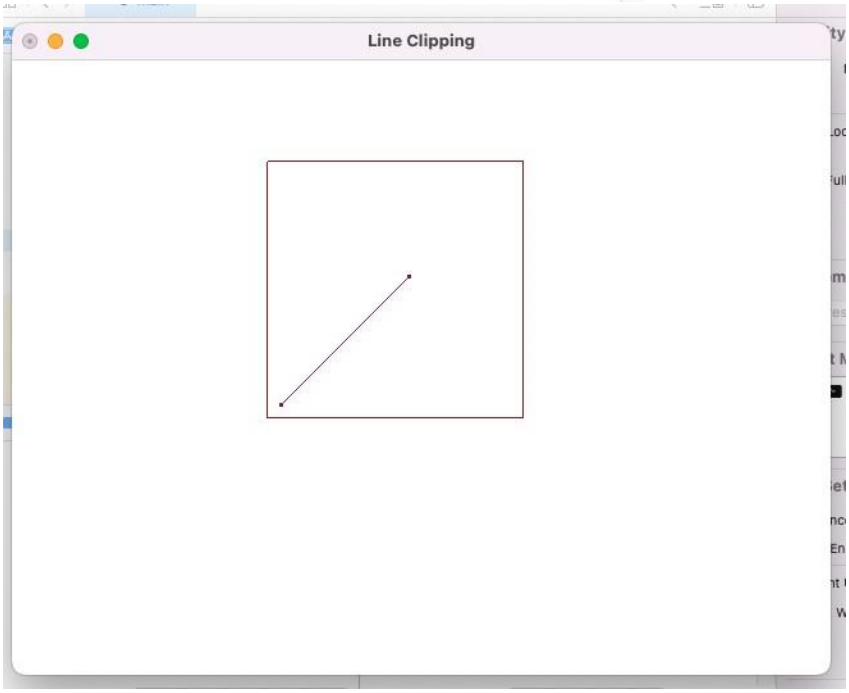
```
glVertex2d(400,200);
glEnd(); glFlush();
glBegin(GL_LINES);
glColor3f(0.5f,0.1f,0.1f);
glVertex2d(400,200);
glVertex2d(200,200);
glEnd(); glFlush();
// P11 = (5, 5), P12 = (7, 7)
glBegin(GL_POINTS);
glVertex2d(x1,y1);
glEnd(); glFlush();
glBegin(GL_POINTS);
glVertex2d(x2,y2);
glEnd(); glFlush();
lineclip(x1,y1,x2,y2); //
Second Line segment
// P21 = (7, 9), P22 = (11, 4)
//lineclip(7,9, 11, 4);;
// Third Line segment
// P31 = (1, 5), P32 = (4, 1)
//lineclip(1, 5, 4, 1); } int
main(int argc,char* argv[]) {
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(640,480);
glutCreateWindow("Line Clipping");
glutDisplayFunc(myDisplay); myInit();
glutMainLoop()
; return 1;
}
```

OUTPUT:

The black square is the clipping window

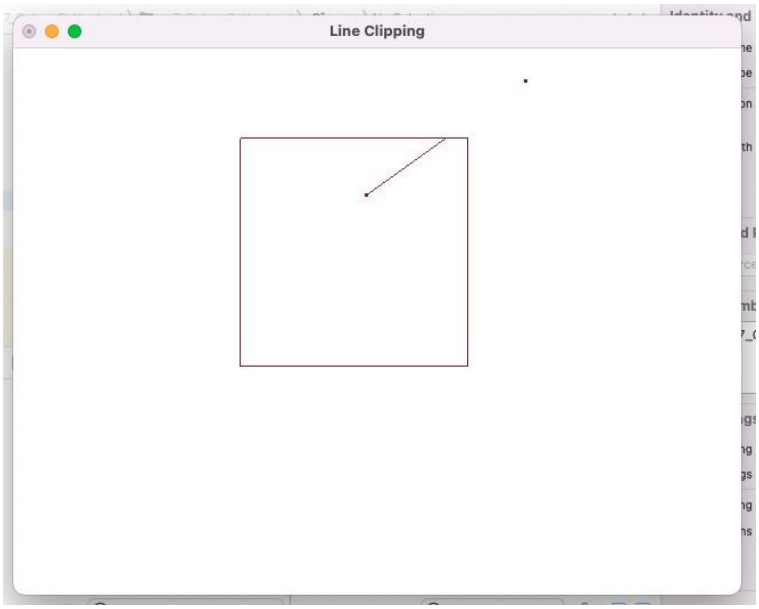
Case 1: Line lies within the clipping window

```
_Cohen Sutherland                    Line: 9  Col: 22
The clipping window has been set with
    the coordinates:
(200,200),(200,400),(400,400) and
    (400,200)

Enter (x1,y1):210 210
Enter (x2,y2):310 310
Line accepted from 210, 210 to 310, 310
```



Case 2: One point lies inside the clipping window

```
ex7_Cohen Sutherland                 Line: 9  Col: 22
The clipping window has been set with
    the coordinates:
(200,200),(200,400),(400,400) and
    (400,200)

Enter (x1,y1):310 350
Enter (x2,y2):450 450
Line accepted from 310, 350 to 380, 400
```



Case 3: Both points lies outside the Clipping window, but lines crosses through the clipping window

```
The clipping window has been set with
    the coordinates:
(200,200),(200,400),(400,400) and
    (400,200)

Enter (x1,y1):110 190
Enter (x2,y2):410 450
Line accepted from 200, 268 to 352.308,
    400
```