MALNAD COLLEGE OF ENGINEERING

(An Autonomous Institute under VTU, Belagavi)

HASSAN – 573202

Subject: Computer Graphics and Visualization **Code:**

18CS602

Submitted by:

Jayshankar

4MC18CS057

6th Sem, B sec

Computer Science

Cohen sutherland line clipping algorithm

```
#include <stdio.h>
#include <GL/glut>
#define outcode int
#define true 1 #define
false 0
double xmin=50,ymin=50, xmax=100,ymax=100; // Window boundaries double
xvmin=200,yvmin=200,xvmax=300,yvmax=300; // Viewport boundaries
//int x1, x2, y1, y2;
//bit codes for the right, left, top, &amp;amp; bottom
const int RIGHT = 8; const int LEFT = 2; const
int TOP = 4; const int BOTTOM = 1;
//used to compute bit codes of a point outcode
ComputeOutCode (double x, double y);
//Cohen-Sutherland clipping algorithm clips a line from
//P0 = (x0, y0) to P1 = (x1, y1) against a rectangle with
//diagonal from (xmin, ymin) to (xmax, ymax).
void CohenSutherlandLineClipAndDraw (double x0, double y0,double x1, double y1)
{
//Outcodes for P0, P1, and whatever point lies outside the clip rectangle
outcode outcode0, outcode1, outcodeOut; bool accept = false, done =
false;
//compute outcodes
outcode0 = ComputeOutCode (x0, y0); outcode1 =
ComputeOutCode (x1, y1); do{ if (!(outcode0 | outcode1)) //logical or
is 0 Trivially accept &amp;amp; exit
{
```

```
accept = true; done
= true;
}
else if (outcode0 &amp;amp; outcode1) //logical and is not 0. Trivially reject and exit done =
true;
else
{
//failed both tests, so calculate the line segment to clip //from
an outside point to an intersection with clip edge double x,
y;
//At least one endpoint is outside the clip rectangle; pick it.
outcodeOut = outcode0? outcode0: outcode1;
//Now find the intersection point;
//use formulas y = y0 + slope * (x - x0), x = x0 + (1/slope)* (y - y0)

if (outcodeOut &amp;amp; TOP) //point is above the clip rectangle
{
x = x0 + (x1 - x0) * (ymax - y0)/(y1 - y0);
y = ymax;
}
else if (outcodeOut &amp;amp; BOTTOM) //point is below the clip rectangle
{
x = x0 + (x1 - x0) * (ymin - y0)/(y1 - y0); y = ymin;
}
else if (outcodeOut &amp;amp; RIGHT) //point is to the right of clip rectangle
{
y = y0 + (y1 - y0) * (xmax - x0)/(x1 - x0);
```

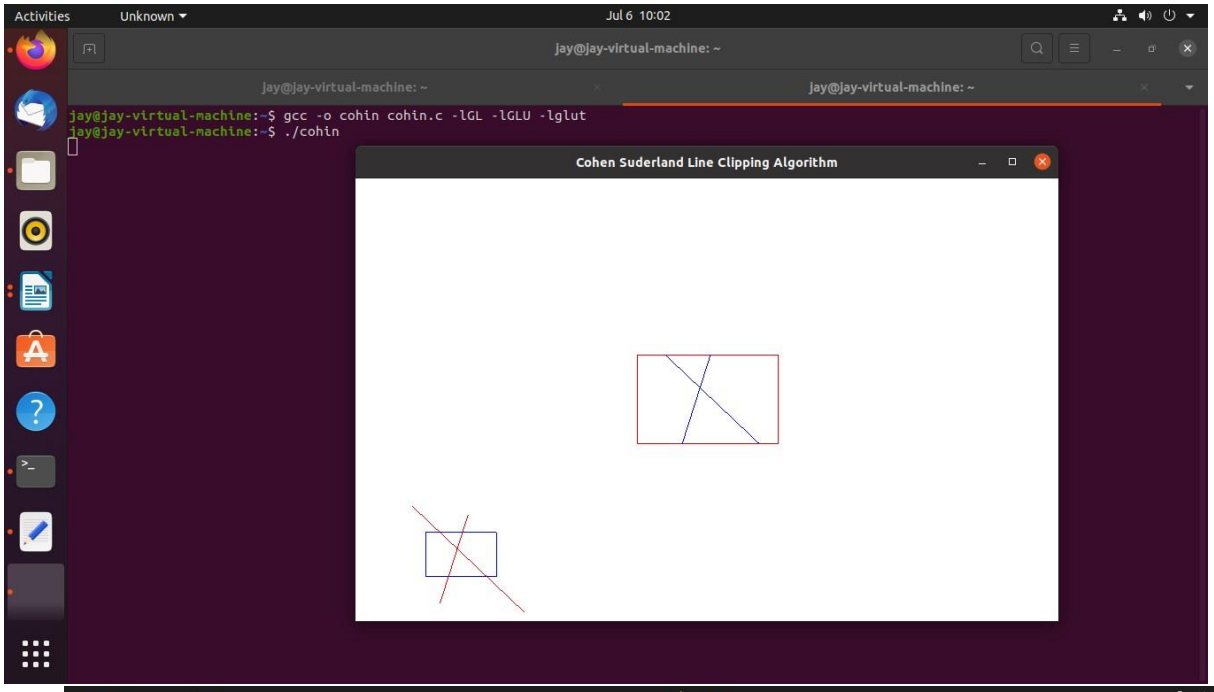
```
x = xmax;
}
else //point is to the left of clip rectangle
{
y = y0 + (y1 - y0) * (xmin - x0)/(x1 - x0); x = xmin;
}
//Now we move outside point to intersection point to clip
//and get ready for next pass. if
(outcodeOut == outcode0)
{
x0 = x; y0
= y;
outcode0 = ComputeOutCode (x0, y0);
} else
{
x1 = x; y1
= y;
outcode1 = ComputeOutCode (x1, y1);
}
}
}while (!done); if
(accept)
{ // Window to viewport mappings double sx=(xvmax-
xvmin)/(xmax-xmin); // Scale parameters double
sy=(yvmax-yvmin)/(ymax-ymin); double vx0=xvmin+(x0-
xmin)*sx; double vy0=yvmin+(y0-ymin)*sy; double
vx1=xvmin+(x1-xmin)*sx; double vy1=yvmin+(y1-
```

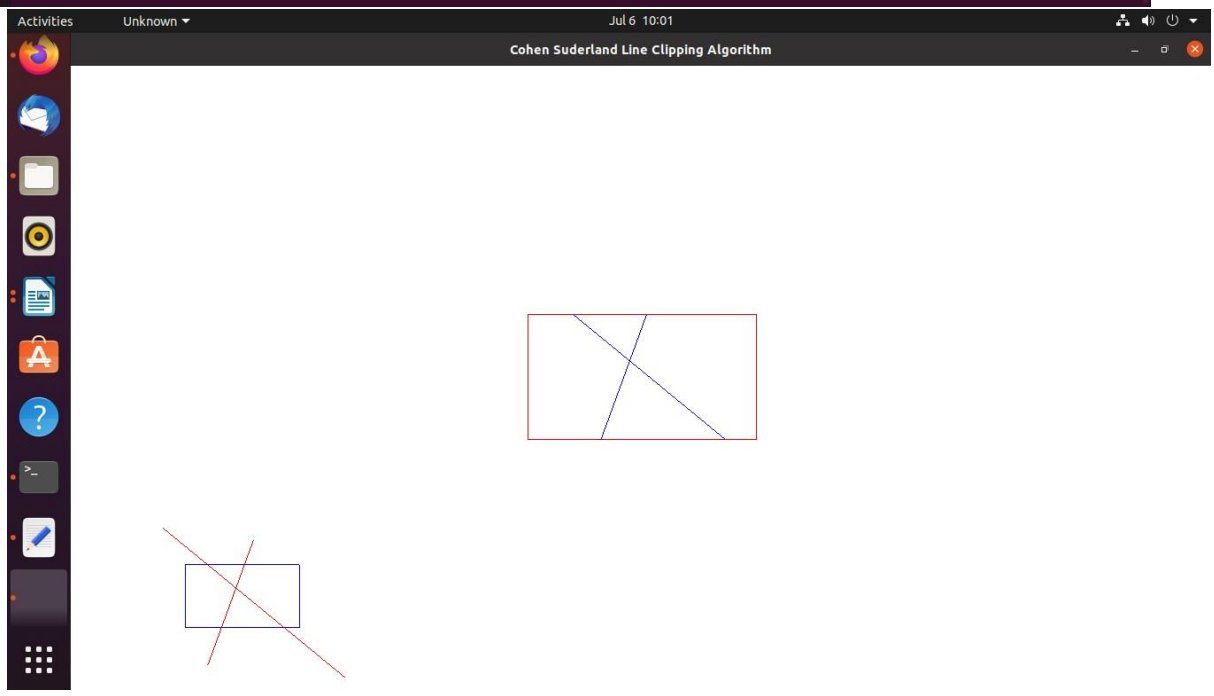
```
ymin)*sy; //draw a red colored viewport glColor3f(1.0, 0.0,
0.0);
glBegin(GL_LINE_LOOP);
glVertex2f(xvmin, yvmin);

glVertex2f(xvmax, yvmin); glVertex2f(xvmax,
yvmax); glVertex2f(xvmin, yvmax); glEnd();
glColor3f(0.0,0.0,1.0); // draw blue colored clipped line
glBegin(GL_LINES);
glVertex2d (vx0, vy0);
glVertex2d (vx1, vy1); glEnd();
}
}
//Compute the bit code for a point (x, y) using the clip rectangle
//bounded diagonally by (xmin, ymin), and (xmax, ymax) outcode
ComputeOutCode (double x, double y)
{
outcode code = 0; if (y &gt; ymax) //above
the clip window code |= TOP; else if (y &lt;
ymin) //below the clip window
code |= BOTTOM;
if (x &gt; xmax) //to the right of clip window
code |= RIGHT;
else if (x &lt; xmin) //to the left of clip window
code |= LEFT; return
code;
}
```

```
void display()
{
double x0=120,y0=10,x1=40,y1=130;
glClear(GL_COLOR_BUFFER_BIT);
//draw the line with red color
glColor3f(1.0,0.0,0.0); //bres(120,20,340,250);
glBegin(GL_LINES);

glVertex2d (x0, y0);
glVertex2d (x1, y1);
glVertex2d (60,20); glVertex2d
(80,120);

glEnd();
//draw a blue colored window
glColor3f(0.0, 0.0, 1.0);
glBegin(GL_LINE_LOOP);
glVertex2f(xmin, ymin); glVertex2f(xmax,
ymin); glVertex2f(xmax, ymax);
glVertex2f(xmin, ymax);

glEnd();
CohenSutherlandLineClipAndDraw(x0,y0,x1,y1);
CohenSutherlandLineClipAndDraw(60,20,80,120);
glFlush();
}
void myinit()
{
```

```
glClearColor(1.0,1.0,1.0,1.0);
glColor3f(1.0,0.0,0.0); glPointSize(1.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity(); gluOrtho2D(0.0,499.0,0.0,499.0);
glMatrixMode(GL_MODELVIEW);
}
int main(int argc, char** argv)
{
//printf(&amp;quot;Enter End points:&amp;quot;);
//scanf(&amp;quot;%d%d%d%d&amp;quot;, &amp;amp;x1,&amp;amp;x2,&amp;amp;y1,&amp;amp;y2);
glutInit(&amp;amp;argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(500,500); glutInitWindowPosition(0,0);
glutCreateWindow(&amp;quot;Cohen Suderland Line Clipping Algorithm&amp;quot;);
glutDisplayFunc(display); myinit();
glutMainLoop();
}
```

Output


Activities
Unknown
Jul 6 10:02
jay@jay-virtual-machine: ~
jay@jay-virtual-machine: ~
jay@jay-virtual-machine: ~
jay@jay-virtual-machine:~$ gcc -o cohin cohin.c -lGL -lGLU -lglut
jay@jay-virtual-machine:~$ ./cohin
Cohen Suderland Line Clipping Algorithm


Activities
Unknown
Jul 6 10:01
Cohen Suderland Line Clipping Algorithm