/\* Cohen-Sutherland Line Clipping Algorithm with Window to viewport Mapping \*/

#include <stdio.h>

#include <GL/gl.h>

#include <GL/glut.h>

#define outcode int

double xmin=50,ymin=50, xmax=100,ymax=100; // Window boundaries

//bit codes for the top,bottom,right & left

const int TOP = 1;

const int BOTTOM = 2;

const int RIGHT = 4;

const int LEFT = 8;

//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 & exit

{

accept = true;

done = true;

}

else if (outcode0 & outcode1) //logical and is not 0. Trivially reject & 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 & TOP) //point is above the clip rectangle

{

x = x0 + (x1 - x0) \* (ymax - y0)/(y1 - y0);

y = ymax;

}

else if (outcodeOut & BOTTOM) //point is below the clip rectangle

{

x = x0 + (x1 - x0) \* (ymin - y0)/(y1 - y0);

y = ymin;

}

else if (outcodeOut & RIGHT)//point is to the right of clip rectangl

{

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 gets 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)

{

//draw a red colored viewport

glColor3f(1.0, 0.0, 0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(xmin+200, ymin+200);

glVertex2f(xmax+200, ymin+200);

glVertex2f(xmax+200, ymax+200);

glVertex2f(xmin+200, ymax+200);

glEnd();

glColor3f(0.0,0.0,1.0); // draw blue colored clipped line

glBegin(GL\_LINES);

glVertex2d (x0+200, y0+200);

glVertex2d (x1+200, y1+200);

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 > ymax) //above the clip window

code |= TOP;

else if (y < ymin) //below the clip window

code |= BOTTOM;

if (x > xmax) //to the right of clip window

code |= RIGHT;

else if (x < 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);

glBegin(GL\_LINES);

glVertex2d (x0, y0);

glVertex2d (x1, y1);

glVertex2d (10,70);

glVertex2d (250,90);

glEnd();

//draw a blue colored window

glColor3f(0.0, 0.0, 1.0);

glBegin(GL\_LINE\_LOOP);

glVertex2d(xmin, ymin);

glVertex2d(xmax, ymin);

glVertex2d(xmax, ymax);

glVertex2d(xmin, ymax);

glEnd();

CohenSutherlandLineClipAndDraw(x0,y0,x1,y1);

CohenSutherlandLineClipAndDraw(10,70,250,90);

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);

}

int main(int argc, char\*\* argv)

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("Cohen Sutherland Line Clipping Algorithm");

glutDisplayFunc(display);

myInit();

glutMainLoop();

return 0;

}