Cohen Sutherland line clipping

```
#include<stdio.h>
#include<GL/glut.h>
#define outcode int
double xmin=50,ymin=50,xmax=100,ymax=100;
double xvmin=200,yvmin=200,xvmax=300,yvmax=300;
const int RIGHT=8;
const int LEFT=2;
const int TOP=4;
const int BOTTOM=1;
outcode computeoutcode(double x,double y);
void cohen_clipping(double x0,double y0,double x1,double y1)
{
        outcode outcode0,outcode1,outcodeout;
        int accept=0,done=0;
        outcode0=computeoutcode(x0,y0);
        outcode1=computeoutcode(x1,y1);
        do
        {
                if(!(outcode0|outcode1))
                {
                        accept=1;
                        done=1;
                }
                else if(outcode0 & outcode1)
                        done=1;
                else
                        {
                        double x,y;
                        outcodeout=outcode0?outcode0:outcode1;

                        if(outcodeout & TOP)
                        {
                                x=x0+(x1-x0)*(ymax-y0)/(y1-y0);
                                y=ymax;
                        }
                        else if(outcodeout & BOTTOM)
                        {
                                x=x0+(x1-x0)*(ymin-y0)/(y1-y0);
                                y=ymin;
                        }
                        else if(outcodeout & RIGHT)
                        {
                                y=y0+(y1-y0)*(xmax-x0)/(x1-x0);
                                x=xmax;
                        }
```

```
                        else
                        {
                                y=y0+(y1-y0)+(xmin-x0)/(x1-x0);
                                x=xmin;
                        }
                        if(outcodeout==outcode0)
                        {
                                x0=x;
                                y0=y;
                                outcode0=computeoutcode(x0,y0);
                        }
                        else
                        {
                                x1=x;
                                y1=y;
                                outcode1=computeoutcode(x1,y1);
                        }
                }
        }while(!done);
        if(accept)
        {
                double sx=(xvmax-xvmin)/(xmax-xmin);
                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;
                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);
                glBegin(GL_LINES);
                glVertex2d(vx0,vy0);
                glVertex2d(vx1,vy1);
                glEnd();
        }
}
outcode computeoutcode(double x,double y)
{
        outcode  code=0;
        if(y>ymax)
                code |=TOP;
```

```
        if(y<ymin)
                code |=BOTTOM;
        if(x>xmax)
                code |=RIGHT;
        if(x<xmin)
                code |=TOP;
         return code;
}
void display()
{
        double x0=120,y0=10,x1=40,y1=130;
        glClear(GL_COLOR_BUFFER_BIT);
        glColor3f(1.0,0.0,0.0);
        glBegin(GL_LINES);
        glVertex2d(x0,y0);
        glVertex2d(x1,y1);
        glVertex2d(60,20);
        glVertex2d(80,120);
        glEnd();
        glColor3f(0.0,0.0,1.0);
        glBegin(GL_LINE_LOOP);
        glVertex2f(xmin,ymin);
        glVertex2f(xmax,ymin);
        glVertex2f(xmax,ymax);
        glVertex2f(xmin,ymax);
        glEnd();
        cohen_clipping(x0,y0,x1,y1);
        cohen_clipping(60,20,80,120);
        glFlush();
}
void init()
{
        glClearColor(1.0,1.0,1.0,1.0);
        glColor3f(1.0,0.0,0.0);
        glPointSize(1.0);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0,640.0,0.0,480.0,1.0,-1.0);
}
int main(int argc,char **argv)
{
        glutInit(&argc,argv);
        glutCreateWindow("welcome");
        glutInitWindowSize(500,500);
        glutInitWindowPosition(0,0);
        glutDisplayFunc(display);
```

```
        init();
        glutMainLoop();
         return 0;
}
```

Polygon clipping

```
#include <GL/glut.h>
struct Point
{
  float x,y;
} w[4],oVer[4];
int Nout;
void drawPoly(Point p[],int n)
{
        glBegin(GL_POLYGON);
        for(int i=0;i<n;i++)
        glVertex2f(p[i].x,p[i].y);
        glEnd();
}
bool insideVer(Point p)
{
        if((p.x>=w[0].x)&&(p.x<=w[2].x))
        if((p.y>=w[0].y)&&(p.y<=w[2].y))
        return true;
        return false;
}
void addVer(Point p)
{
        oVer[Nout]=p;
        Nout=Nout+1;
}
Point getInterSect(Point s,Point p,int edge)
{
        Point in;
        float m;
        if(w[edge].x==w[(edge+1)%4].x){ //Vertical Line
                m=(p.y-s.y)/(p.x-s.x);
                in.x=w[edge].x;
                in.y=in.x*m+s.y;
        }
        else
       {
m=(p.y-s.y)/(p.x-s.x);
```

```
                in.y=w[edge].y;
                in.x=(in.y-s.y)/m;
        }
        return in;
}
void clipAndDraw(Point inVer[],int Nin)
{
        Point s,p,interSec;
        for(int i=0;i<4;i++)
        {
                Nout=0;
                s=inVer[Nin-1];
                for(int j=0;j<Nin;j++)
                {
                        p=inVer[j];
                        if(insideVer(p)==true){
                                if(insideVer(s)==true){
                                        addVer(p);
                                }
                                else{
                                        interSec=getInterSect(s,p,i);
                                        addVer(interSec);
                                        addVer(p);
                                }
                        }
                        else{
                                if(insideVer(s)==true){
                                        interSec=getInterSect(s,p,i);
                                        addVer(interSec);
                                }
                        }
                        s=p;
                }
                inVer=oVer;
                Nin=Nout;
        }
        drawPoly(oVer,4);
}
void init()
{
        glClearColor(0.0f,0.0f,0.0f,0.0f);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(0.0,100.0,0.0,100.0,0.0,100.0);
        glClear(GL_COLOR_BUFFER_BIT);
        w[0].x =15,w[0].y=10;
```

```
	w[1].x =15,w[1].y=40;
	w[2].x =40,w[2].y=40;
	w[3].x =40,w[3].y=10;
}
void display(void)
{
	Point inVer[4];
	init();
	glColor3f(1.0f,1.0f,0.0f);
	drawPoly(w,4);
	glColor3f(0.0f,1.0f,0.0f);
	inVer[0].x =10,inVer[0].y=40;
	inVer[1].x =10,inVer[1].y=30;
	inVer[2].x =30,inVer[2].y=30;
	inVer[3].x =30,inVer[3].y=40;
	drawPoly(inVer,4);
	glColor3f(0.0f,0.0f,1.0f);
	clipAndDraw(inVer,4);
	glFlush();
}
int main(int argc,char *argv[])
{
	glutInit(&argc,argv);
	glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
	glutInitWindowSize(400,400);
	glutInitWindowPosition(100,100);
	glutCreateWindow("Polygon Clipping!");
	glutDisplayFunc(display);
	glutMainLoop();
	return 0;
}
```