**clipping**

#include <stdio.h>

#include<iostream>

#include<GL/glut.h>

#include<math.h>

#include<bits/stdc++.h>

using namespace std ;

int result;

int xmin, ymin, xmax, ymax, pt[30][2], w[30][2],n=0,flg=0;

int leftClip(int limit, int xm)

{

int i, j = 0, x1, y1, x2, y2;

float m;

for (i = 0; i < limit; i++)

{

x1 = pt[i][0];

y1 = pt[i][1];

x2 = pt[(i + 1) % limit][0];

y2 = pt[(i + 1) % limit][1];

if (x2 - x1)

m = (y2 - y1) \* 1.0 / (x2 - x1);

if (x1 < xm && x2 < xm)

continue;

if (x1 > xm && x2 > xm)

{

w[j][0] = x2;

w[j++][1] = y2;

continue;

}

if (x1 > xm && x2 < xm)

{

w[j][0] = xm;

w[j++][1] = y1 + m \* (xm - x1);

continue;

}

if (x1 < xm && x2 > xm)

{

w[j][0] = xm;

w[j++][1] = y1 + m \* (xm - x1);

w[j][0] = x2;

w[j++][1] = y2;

}

}

for (i = 0; i < j; i++)

{

pt[i][0] = w[i][0];

pt[i][1] = w[i][1];

w[i][0] = w[i][1] = 0;

}

if (j < limit)

for (; i < limit; i++)

pt[i][0] = pt[i][1] = 0;

return j;

}

int topClip(int limit, int ym)

{

int i, j = 0, x1, y1, x2, y2;

float m;

for (i = 0; i < limit; i++)

{

x1 = pt[i][0];

y1 = pt[i][1];

x2 = pt[(i + 1) % limit][0];

y2 = pt[(i + 1) % limit][1];

if (x2 - x1)

m = (y2 - y1) \* 1.0 / (x2 - x1);

if (y1 < ym && y2 < ym)

continue;

if (y1 > ym && y2 > ym)

{

w[j][0] = x2;

w[j++][1] = y2;

continue;

}

if (y1 > ym && y2 < ym)

{

w[j][0] = x1 + (ym - y1) / m;

w[j++][1] = ym;

continue;

}

if (y1 < ym && y2 > ym)

{

w[j][0] = x1 + (ym - y1) / m;

w[j++][1] = ym;

w[j][0] = x2;

w[j++][1] = y2;

}

}

for (i = 0; i < j; i++)

{

pt[i][0] = w[i][0];

pt[i][1] = w[i][1];

w[i][0] = w[i][1] = 0;

}

if (j < limit)

for (; i < limit; i++)

pt[i][0] = pt[i][1] = 0;

return j;

}

int rightClip(int limit, int xm)

{

int i, j = 0, x1, y1, x2, y2;

float m;

for (i = 0; i < limit; i++)

{

x1 = pt[i][0];

y1 = pt[i][1];

x2 = pt[(i + 1) % limit][0];

y2 = pt[(i + 1) % limit][1];

if (x2 - x1)

m = (y2 - y1) \* 1.0 / (x2 - x1);

if (x1 > xm && x2 > xm)

continue;

if (x1 < xm && x2 < xm)

{

w[j][0] = x2;

w[j++][1] = y2;

continue;

}

if (x1 < xm && x2 > xm)

{

w[j][0] = xm;

w[j++][1] = y1 + m \* (xm - x1);

continue;

}

if (x1 > xm && x2 < xm)

{

w[j][0] = xm;

w[j++][1] = y1 + m \* (xm - x1);

w[j][0] = x2;

w[j++][1] = y2;

}

}

for (i = 0; i < j; i++)

{

pt[i][0] = w[i][0];

pt[i][1] = w[i][1];

w[i][0] = w[i][1] = 0;

}

if (j < limit)

for (; i < limit; i++)

pt[i][0] = pt[i][1] = 0;

return j;

}

int bottomClip(int limit, int ym)

{

int i, j = 0, x1, y1, x2, y2;

float m;

for (i = 0; i < limit; i++)

{

x1 = pt[i][0];

y1 = pt[i][1];

x2 = pt[(i + 1) % limit][0];

y2 = pt[(i + 1) % limit][1];

if (x2 - x1)

m = (y2 - y1) \* 1.0 / (x2 - x1);

if (y1 > ym && y2 > ym)

continue;

if (y1 < ym && y2 < ym)

{

w[j][0] = x2;

w[j++][1] = y2;

continue;

}

if (y1 < ym && y2 > ym)

{

w[j][0] = x1 + (ym - y1) / m;

w[j++][1] = ym;

continue;

}

if (y1 > ym && y2 < ym)

{

w[j][0] = x1 + (ym - y1) / m;

w[j++][1] = ym;

w[j][0] = x2;

w[j++][1] = y2;

}

}

for (i = 0; i < j; i++)

{

pt[i][0] = w[i][0];

pt[i][1] = w[i][1];

w[i][0] = w[i][1] = 0;

}

if (j < limit)

for (; i < limit; i++)

pt[i][0] = pt[i][1] = 0;

return j;

}

void display(void)

{

}

void init()

{

glClearColor(0.0,0.0,0.0,0.0);

glClear (GL\_COLOR\_BUFFER\_BIT);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,0.0,0.0);

glPointSize(2.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,700,0,700) ;

}

void menu(int c)

{

if(c==1)

{

result = leftClip(n, xmin);

result = topClip(result, ymin);

result = rightClip(result, xmax);

result = bottomClip(result, ymax);

}

if(c==2)

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(0.0,0.0,0.0);

glBegin(GL\_POINTS);

glVertex2i(0,0);

glEnd();

glFlush();

glColor3f(1.0,1.0,0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(xmin,ymin);

glVertex2i(xmax,ymin);

glVertex2i(xmax,ymax);

glVertex2i(xmin,ymax);

glEnd();

glFlush();

for (int i = 0; i < result; i++)

{

glColor3f(0.0,0.0,1.0);

glBegin(GL\_LINE\_STRIP);

glVertex2i(pt[i][0],pt[i][1]);

glVertex2i(pt[(i+1)%result][0],pt[(i+1)%result][1]);

glEnd();

glFlush();

} } }

void mouse(int button, int state, int cx, int cy )

{

if(state==GLUT\_DOWN)

{

if(button==GLUT\_LEFT\_BUTTON)

{

pt[n][0] = cx ;

pt[n][1] = 700-cy ;

n++;

if(n>1)

{

glColor3f(1.0,0.0,0.0);

glBegin(GL\_LINE\_STRIP);

glVertex2i(pt[n-2][0],pt[n-2][1]);

glVertex2i(pt[n-1][0],pt[n-1][1]);

glEnd();

glFlush();

} }

if(button==GLUT\_RIGHT\_BUTTON)

{ if(flg==0)

{

cout<<"RIGHT CLICK 1 - DONE"<<endl;

xmin=cx;

ymin=700-cy;

flg++;

} else

{

cout<<"RIGHT CLICK 2 - DONE"<<endl;

xmax=cx;

ymax=700-cy;

glColor3f(1.0,1.0,0.0);

glBegin(GL\_LINE\_LOOP);

glVertex2i(xmin,ymin);

glVertex2i(xmax,ymin);

glVertex2i(xmax,ymax);

glVertex2i(xmin,ymax);

glEnd();

glFlush();

} } }}

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

{

glutInit(&argc,argv);

glutInitWindowSize(700,700);

glutInitWindowPosition(500,50);

glutCreateWindow(" COHEN SUTHERLAND POLYGON CLIPPING ");

cout<<"PLEASE FOLLOW THESE STEPS:"<<endl;

cout<<"[1] MAKE POLYGON by USING LEFT BUTTON CLICK"<<endl;

cout<<"[2] SELECT WINDOW COORDINATES by USING RIGHT BUTTON CLICK: where"<<endl;

cout<<"FIRST RIGHT CLICK - SELECT Xmin and Ymin"<<endl;

cout<<"SECOND RIGHT CLICK - SELECT Xmax and Ymax"<<endl<<endl;

cout<<"[3]In MENU, please first execute and then clip"<<endl<<endl;

init();

glutMouseFunc(mouse);

glutDisplayFunc(display);

glutCreateMenu(menu);

glutAddMenuEntry("EXECUTE",1);

glutAddMenuEntry("SHOW CLIPPED",2);

glutAttachMenu(GLUT\_MIDDLE\_BUTTON);

glutMainLoop(); return 0; }