1. **Midpoint\_line**

#include<GL/glut.h>

#include<stdlib.h>

#include<stdio.h>

int x0,y0,x1,y1;

void init()

{

glClearColor(1,1,1,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

// glMatrixMode(GL\_PROJECTION);

gluOrtho2D(-500,500,-500,500);

}

void writepixel(int x,int y)

{

glPointSize(1);

glBegin(GL\_POINTS);

glVertex2f(x,y);

glColor3f(0,0,0);

glEnd();

glFlush();

}

void display()

{

//int i,j;

float dx=x1-x0,dy=y1-y0;

float incre=dy,incrne=dy-dx,d=2\*dy-dx;

int x=x0;

int y=y0;

writepixel(x,y);

if(abs(dy/dx)<=1){

while(x<x1)

{

if(d<=0)

{

d+=2\*incre;

x+=1;

}

else

{

d+=2\*incrne;

x+=1;

y+=1;

}

writepixel(x,y);

}

}

else

{

d=2\*dx-dy;

incre=dx;

incrne=dx-dy;

while(y<y1)

{

y=y+1;

if(d<=0)

d+=2\*incre;

else

{

d+=2\*incrne;

x=x+1;

}

writepixel(x,y);

}

}

}

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

{

glutInit(&argc,argv);

printf("Enter the co-ordinates of two points\n\n");

printf("X0="); scanf("%d",&x0);

printf("\nY0="); scanf("%d",&y0);

printf("\nX1="); scanf("%d",&x1);

printf("\nY1="); scanf("%d",&y1);

/\* cout<<"Enter the co-ordinates of two points\n\n";

cout<<"X0=";

cin>>x0;

cout<<"\nY0=";

cin>>y0;

cout<<"\nX1=";

cin>>x1;

cout<<"\nY1=";

cin>>y1;\*/

glutInitWindowSize(1000,1000);

glutCreateWindow("Midpoint Point line Algorithm");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

1. **Midpoint circle**

#include<stdlib.h>

#include<stdio.h>

#include<GL/glut.h>

float r;

void init()

{

glClearColor(1,0,0,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

gluOrtho2D(-250,500,-250,500);

}

void writepixel(int x,int y)

{

glPointSize(1);

glBegin(GL\_POINTS);

glVertex2d(x,y);

glColor3f(1,1,1);

glEnd();

glFlush();

}

void display()

{

int x=0,y=r;

double d=5.0/4.0-r;

writepixel(x,y);

while(x<y)

{

if(d<0)

{

d+=2.0\*x+3.0;

}

else

{

d+=2.0\*(x-y)+5.0;

y--;

}

x++;

writepixel(x,y);

writepixel(y,x);

writepixel(x,-y);

writepixel(y,-x);

writepixel(-x,y);

writepixel(-y,x);

writepixel(-x,-y);

writepixel(-y,-x);

}

}

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

{

printf("Enter the radius of the circle:-");

scanf("%f",&r);

glutInit(&argc,argv);

glutInitWindowSize(500,500);

glutCreateWindow("Midpoint circle algorithm");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

1. **Cohen\_sutherland line clipping**

#include<stdio.h>

#include<GL/glut.h>

float xmin,ymin,xmax,ymax,lx0,ly0,lx1,ly1;

int TOP=8,BOTTOM=4,RIGHT=2,LEFT=1;

void display();

void init()

{

glClearColor(0,0,0,0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(-100,500,-100,500);

}

int getOutCode(float x,float y)

{

int c=0;

if(y>ymax)

c=TOP;

if(y<ymin)

c=BOTTOM;

if(x>xmax)

c=c|RIGHT;

if(x<xmin)

c=c|LEFT;

return c;

}

void lineclip(float x1,float y1,float x2,float y2)

{

int outcode1=getOutCode(x1,y1);

int outcode2=getOutCode(x2,y2);

float m=(y2-y1)/(x2-x1);

while((outcode1|outcode2)!=0)

{

printf("%d",outcode1);

if((outcode1&outcode2)!=0)

{

lx0=lx1=ly0=ly1=-500;

break;

}

float xi=x1,yi=y1;

int c=outcode1;

if(c==0){

xi=x2;

yi=y2;

c=outcode2;

}

float x,y;

if((c&TOP)!=0)

{

y=ymax;

x=xi+1.0/m\*(ymax-yi);

}

else if((c&BOTTOM)!=0)

{

y=ymin;

x=xi+1.0/m\*(ymin-yi);

}

else if((c&RIGHT)!=0)

{

x=xmax;

y=yi+m\*(xmax-xi);

}

else if((c&LEFT)!=0)

{

x=xmin;

y=yi+m\*(xmin-xi);

}

if(c==outcode1)

{

lx0=x;

ly0=y;

outcode1=getOutCode(lx0,ly0);

}

if(c==outcode2)

{

lx1=x;

ly1=y;

outcode2=getOutCode(lx1,ly1);

}

}

display();

}

void keypress(unsigned char key,int x,int y)

{

if(key=='c')

{

lineclip(lx0,ly0,lx1,ly1);

printf("Line clipped");

glFlush();

}

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(xmin,ymin);

glVertex2f(xmin,ymax);

glVertex2f(xmax,ymax);

glVertex2f(xmax,ymin);

glEnd();

glColor3f(0,1,0);

glBegin(GL\_LINES);

glVertex2f(lx0,ly0);

glVertex2f(lx1,ly1);

glEnd();

glFlush();

}

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

printf("(Line clipping window parameters format xmin,ymin,xmax,ymax)");

printf("\nEnter the window parameters\n");

scanf("%f%f%f%f",&xmin,&ymin,&xmax,&ymax);

printf("\n(Line co-ordinates format x0,y0,x1,y1)");

printf("\nEnter the line co-ordinates\n");

scanf("%f%f%f%f",&lx0,&ly0,&lx1,&ly1);

glutInit(&argc,argv);

//glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(700,700);

glutInitWindowPosition(0,0);

glutCreateWindow("Line Clip");

glutDisplayFunc(display);

glutKeyboardFunc(keypress);

init();

glutMainLoop();

return 0;

}

1. **Liang\_barsky line clipping**

#include<stdio.h>

#include<GL/glut.h>

int xmin=-100,ymin=-100,xmax=100,ymax=100;

float t1=0,t2=1;

float p[4],q[4];

int x01,y01,x02,y02,x03,y03,x04,y04;

void init()

{

glClearColor(0,0,0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(-300,300,-300,300);

}

void display()

{

glBegin(GL\_LINE\_LOOP);

glColor3f(0,1,0);

glVertex2f(xmin,ymin);

glVertex2f(xmin,ymax);

glVertex2f(xmax,ymax);

glVertex2f(xmax,ymin);

glEnd();

glBegin(GL\_LINES);

glVertex2f(x01,y01);

glVertex2f(x02,y02);

glEnd();

glFlush();

}

void lineclip()

{

int dx,dy;

dx=x02-x01;

dy=y02-y01;

p[0]=-dx;

p[1]=dx;

p[2]=-dy;

p[3]=dy;

q[0]=x01-xmin;

q[1]=xmax-x01;

q[2]=y01-ymin;

q[3]=ymax-y01;

int i;

float t;

int f=0;

for(i=0;i<4;i++)

{

if(p[i]==0)

{

if(q[i]<0)

return;

}

if(p[i]<0)

{

t=q[i]/p[i];

if(t>t1 && t<t2)

t1=t;

}

else if(p[i]>0)

{

t=q[i]/p[i];

if(t>t1 && t<t2)

t2=t;

}

/\*if(q[i]<0)

{

x01=y01=x02=y02=0;

}\*/

}

if(t1<t2)

{

x03=x01+t1\*(x02-x01);

y03=y01+t1\*(y02-y01);

x04=x01+t2\*(x02-x01);

y04=y01+t2\*(y02-y01);

if(x03>=xmin&&x03<=xmax&&y03>=ymin&&y03<=ymax){

if(x04>=xmin&&x04<=xmax&&y04>=ymin&&y04<=ymax){

glBegin(GL\_LINES);

glColor3f(1,1,1);

glVertex2f(x03,y03);

glVertex2f(x04,y04);

glEnd();

glFlush();}

}

}

}

void keypressed(unsigned char key,int x,int y)

{

if(key=='c')

{

glClearColor(0,0,0,0);

glClear(GL\_COLOR\_BUFFER\_BIT);

glBegin(GL\_LINE\_LOOP);

glColor3f(0,1,0);

glVertex2f(xmin,ymin);

glVertex2f(xmin,ymax);

glVertex2f(xmax,ymax);

glVertex2f(xmax,ymin);

glEnd();

glFlush();

lineclip();

}

}

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

{

printf("Enter the line co-ordinates\n");

scanf("%d%d%d%d",&x01,&y01,&x02,&y02);

glutInit(&argc,argv);

glutInitWindowSize(1000,1000);

glutCreateWindow("Liang\_barsky");

init();

glutDisplayFunc(display);

glutKeyboardFunc(keypressed);

glutMainLoop();

}

1. **Rectangular\_mesh**

#include<stdio.h>

#include<GL/glut.h>

#define maxx 40

#define maxy 10

#define dx 20

#define dy 20

float x[maxx+1],y[maxy+1],x0=100,y0=100;

int i,j;

void init()

{

glClearColor(0,0,0,0);

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,500,0,700);

}

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

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

x[i]=x0+i\*dx;

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

y[j]=y0+j\*dy;

//glColor3f(1,0,1);

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

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

{

glBegin(GL\_LINE\_LOOP);

glVertex2f(x[i],y[j]);

glVertex2f(x[i],y[j+1]);

glVertex2f(x[i+1],y[j+1]);

glVertex2f(x[i+1],y[j]);

glEnd();

}

glFlush();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(700,500);

glutCreateWindow("Polygon mesh");

init();

glutDisplayFunc(display);

glutMainLoop();

return 0;

}

1. **House Rotation**

#include<GL/glut.h>

#include<stdio.h>

#include<math.h>

float house[2][9]={{100.0,100.0,175.0,250.0,250.0,150.0,150.0,200.0,200.0},{100.0,300.0,400.0,300.0,100.0,100.0,150.0,150.0,100.0}};

float theta;

float h=175.0,k=400.0;

void drawhouse()

{

glColor3f(1,0,0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(house[0][0],house[1][0]);

glVertex2f(house[0][1],house[1][1]);

glVertex2f(house[0][3],house[1][3]);

glVertex2f(house[0][4],house[1][4]);

glEnd();

glColor3f(1,0,1);

glBegin(GL\_LINE\_LOOP);

glVertex2f(house[0][5],house[1][5]);

glVertex2f(house[0][6],house[1][6]);

glVertex2f(house[0][7],house[1][7]);

glVertex2f(house[0][8],house[1][8]);

glEnd();

glColor3f(1,0,0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(house[0][1],house[1][1]);

glVertex2f(house[0][2],house[1][2]);

glVertex2f(house[0][3],house[1][3]);

glEnd();

glFlush();

}

void display()

{

int i;

float m[16],p,q;

p=-h\*(cos(theta)-1)+k\*(sin(theta));

q=-k\*(cos(theta)-1)-h\*(sin(theta));

for(i=0;i<15;i++)

m[i]=0.0;

m[0]=cos(theta);

m[1]=sin(theta);

m[4]=-sin(theta);

m[5]=cos(theta);

m[12]=p;

m[13]=q;

m[10]=1;

m[15]=1;

//glMatrixMode(GL\_MODELVIEW);

glClear(GL\_COLOR\_BUFFER\_BIT);

drawhouse();

glPushMatrix();

glMultMatrixf(m);

drawhouse();

glPopMatrix();

glFlush();

}

void init()

{

glClearColor(1,1,1,1);

//glMatrixMode(GL\_PROJECTION);

glColor3f(1,1,0);

//glLoadIdentity();

gluOrtho2D(0,499,0,499);

// glMatrixMode(GL\_MODELVIEW);

}

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

{

printf("Enter the angle of rotation\n");

scanf("%f",&theta);

theta=(theta\*3.14)/180;

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

glutInitWindowSize(500,500);

glutInitWindowPosition(0,0);

glutCreateWindow("House Rotation");

glutDisplayFunc(display);

init();

glutMainLoop();

return 0;

}