## **Prog** 1

#include "stdafx.h"

#include<glut.h>

#include<stdio.h>

int x1,y1,x2,y2;

void myInit()

{

    glClearColor(1.0,1.0,1.0,0.0);

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glPointSize(3.0);

    glMatrixMode(GL\_PROJECTION);

    gluOrtho2D(0,200,0,200);

}

void draw\_pixel(int x, int y)

{

    glBegin(GL\_POINTS);

    glColor3f(1.0,0.0,0.0);

    glVertex2i(x,y);

    glEnd();

}

void draw\_line(int x1,int x2, int y1, int y2)

{

    int dx,dy,i,e;

    int incx, incy, inc1, inc2;

    int x,y;

    dx=x2-x1;

    dy=y2-y1;

    if(dx<0)dx=-dx;

    if(dy<0)dy=-dy;

    incx=1;

    if(x2<x1)incx=-1;

    incy=1;

    if(y2<y1)incy=-1;

    x=x1;

    y=y1;

    if(dx>dy)

    {

        draw\_pixel(x,y+10);

        e=2\*dy-dx;

        inc1=2\*(dy-dx);

        inc2=2\*dy;

        for(i=0;i<dx;i++)

        {

            if(e>=0)

            {

                y+=incy;

                e+=inc1;

            }

            else

                e+=inc2;

                x+=incx;

                draw\_pixel(x,y+10);

        }

    }

    else

    {

        draw\_pixel(x,y+10);

        e=2\*dx-dy;

        inc1=2\*(dx-dy);

        inc2=2\*dx;

        for(i=0;i<dy;i++)

        {

            if(e>=0)

            {

                x+=incx;

                e+=inc1;

            }

            else

                e+=inc2;

                y+=incy;

                draw\_pixel(x,y+10);

        }

    }

}

void myDisplay()

{

    draw\_line(x1,x2,y1,y2);

    glFlush();

}

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

{

    printf("enter(x1,y1,x2,y2)\n");

    scanf\_s("%d %d %d %d", &x1,&y1,&x2,&y2);

    glutInit(&argc,argv);

    glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

    glutInitWindowSize(500,500);

    glutInitWindowPosition(10,10);

    glutCreateWindow(" ...............ABC...............Bresenham's line drawing");

    myInit();

    glutDisplayFunc(myDisplay);

    glutMainLoop();

}

## **Prog 2**

#include "stdafx.h"

#include <glut.h>

float x,y;

int angle;

void init()

{

    glClearColor(0.0,0.0,0.0,0.0);

    gluOrtho2D(-100,100,-100,100);

}

void triangle()

{

    glBegin(GL\_TRIANGLES);

        glVertex2f(10,10);

        glVertex2f(30,10);

        glVertex2f(20,40);

        glEnd();

}

void display ()

{

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(1.0,1.0,1.0);

    glBegin(GL\_LINES);

    glVertex2f(-100,0);

    glVertex2f(100,0);

    glVertex2f(0,-100);

    glVertex2f(0,100);

    glEnd();

    glColor3f(1.0,0.0,0.0);

    triangle();

    glPushMatrix();

    glRotatef(angle,0.0,0.0,1.0);

    glColor3f(0.0,1.0,0.0);

    triangle();

    glPopMatrix();

    glPushMatrix();

    glTranslatef(x,y,0.0);

    glRotatef(angle,0.0,0.0,1);

    glTranslatef(-x,-y,0.0);

    glColor3f(0.0,0.0,1.0);

    triangle();

    glPopMatrix();

    glFlush();

}

int main()

{

    printf("angle\n");

    scanf\_s("%d",&angle);

    printf("fixed points X and Y \n");

    scanf\_s("%f%f",&x,&y);

    glutInitDisplayMode(GLUT\_SINGLE);

    glutInitWindowSize(600,600);

    glutCreateWindow("Triangle rotation");

    init();

    glutDisplayFunc(display);

    glutMainLoop();

}

## **Prog 3**

#include "stdafx.h"

#include <glut.h>

GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0}, {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0}, {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};

GLfloat normals[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0}, {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-1.0,1.0},{1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};

GLfloat colors[][3] = {{0.0,0.0,0.0},{1.0,0.0,0.0}, {1.0,1.0,0.0}, {0.0,1.0,0.0},{0.0,0.0,1.0}, {1.0,0.0,1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0}};

void polygon(int a, int b, int c , int d)

{

glBegin(GL\_POLYGON);

glColor3fv(colors[a]);

glNormal3fv(normals[a]);

glVertex3fv(vertices[a]);

glColor3fv(colors[b]);

glNormal3fv(normals[b]);

glVertex3fv(vertices[b]);

glColor3fv(colors[c]);

glNormal3fv(normals[c]);

glVertex3fv(vertices[c]);

glColor3fv(colors[d]);

glNormal3fv(normals[d]);

glVertex3fv(vertices[d]);

glEnd();

}

void colorcube(void)

{

polygon(0,3,2,1);

polygon(2,3,7,6);

polygon(0,4,7,3);

polygon(1,2,6,5);

polygon(4,5,6,7);

polygon(0,1,5,4);

}

static GLfloat theta[] =

{0.0,0.0,0.0}; static GLint axis = 2;

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT |

GL\_DEPTH\_BUFFER\_BIT); glLoadIdentity();

glRotatef(theta[0], 1.0, 0.0, 0.0);

glRotatef(theta[1], 0.0, 1.0, 0.0);

glRotatef(theta[2], 0.0, 0.0, 1.0);

colorcube();

glFlush();

glutSwapBuffers();

}

void spinCube()

{

theta[axis] += 0.1;

if( theta[axis] > 360.0 ) theta[axis] -= 360.0;

glutPostRedisplay();

}

void mouse(int btn, int state, int x, int y)

{

if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) axis = 0;

if(btn==GLUT\_MIDDLE\_BUTTON && state == GLUT\_DOWN) axis = 1;

if(btn==GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN) axis = 2;

}

void myReshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

glOrtho(-2.0, 2.0, -2.0 \* (GLfloat) h / (GLfloat) w, 2.0 \* (GLfloat) h / (GLfloat) w, -10.0, 10.0);

else

glOrtho(-2.0 \* (GLfloat) w / (GLfloat) h, 2.0 \* (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH);

glutInitWindowSize(500, 500);

glutCreateWindow("Rotating a Color Cube");

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

glutIdleFunc(spinCube);

glutMouseFunc(mouse);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

}

## **Prog 4**

#include "stdafx.h"

#include <stdlib.h>

#include <glut.h>

GLfloat vertices[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0}, {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-

1.0,1.0}, {1.0,-1.0,1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};

GLfloat normals[][3] = {{-1.0,-1.0,-1.0},{1.0,-1.0,-1.0}, {1.0,1.0,-1.0}, {-1.0,1.0,-1.0}, {-1.0,-

1.0,1.0}, {1.0,-1.0, 1.0}, {1.0,1.0,1.0}, {-1.0,1.0,1.0}};

GLfloat colors[][3] = {{0.0,0.0,0.0},{1.0,0.0,0.0}, {1.0,1.0,0.0}, {0.0,1.0,0.0},

{0.0,0.0,1.0}, {1.0, 0.0, 1.0}, {1.0,1.0,1.0}, {0.0,1.0,1.0}};

void polygon(int a, int b, int c , int d)

{

glBegin(GL\_POLYGON);

glColor3fv(colors[a]);

glNormal3fv(normals[a]);

glVertex3fv(vertices[a]);

glColor3fv(colors[b]);

glNormal3fv(normals[b]);

glVertex3fv(vertices[b]);

glColor3fv(colors[c]);

glNormal3fv(normals[c]);

glVertex3fv(vertices[c]);

glColor3fv(colors[d]);

glNormal3fv(normals[d]);

glVertex3fv(vertices[d]);

glEnd();

}

void colorcube()

{

polygon(0,3,2,1);

polygon(2,3,7,6);

polygon(0,4,7,3);

polygon(1,2,6,5);

polygon(4,5,6,7);

polygon(0,1,5,4);

}

static GLfloat theta[] =

{0.0,0.0,0.0}; static GLint axis = 2;

static GLdouble viewer[]= {0.0, 0.0, 5.0};

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glLoadIdentity();

gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

glRotatef(theta[0], 1.0, 0.0,0.0);

glRotatef(theta[1], 0.0,1.0, 0.0);

glRotatef(theta[2],0.0, 0.0, 1.0);

colorcube();

glFlush();

glutSwapBuffers();

}

void mouse(int btn, int state, int x, int y)

{

if(btn==GLUT\_LEFT\_BUTTON && state == GLUT\_DOWN) axis = 0;

if(btn==GLUT\_MIDDLE\_BUTTON && state == GLUT\_DOWN) axis = 1;

if(btn==GLUT\_RIGHT\_BUTTON && state == GLUT\_DOWN) axis = 2;

theta[axis] += 2.0;

if( theta[axis] > 360.0 )

theta[axis] -=360.0;

display();

}

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

{

if(key == 'x') viewer[0]-= 1.0;

if(key == 'X') viewer[0]+=

1.0; if(key == 'y') viewer[1]-=

1.0; if(key == 'Y')

viewer[1]+= 1.0; if(key == 'z')

viewer[2]-= 1.0; if(key == 'Z')

viewer[2]+= 1.0; display();

}

void myReshape(int w, int h)

{

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if(w<=h)

glFrustum(-2.0, 2.0, -2.0 \* (GLfloat) h/ (GLfloat) w, 2.0\* (GLfloat) h / (GLfloat) w, 2.0, 20.0);

else

glFrustum(-2.0, 2.0, -2.0 \* (GLfloat) w/ (GLfloat) h, 2.0\* (GLfloat) w / (GLfloat) h, 2.0, 20.0);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB |

GLUT\_DEPTH);

glutInitWindowSize(500, 500);

glutCreateWindow("Colorcube Viewer");

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

glutMouseFunc(mouse);

glutKeyboardFunc(keys);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();}

## **Prog 5**

#include<stdio.h>

#include<math.h>

#include "stdafx.h"

#include<glut.h>

int flag = 0;

#define outcode int

float  xmin = 200, xmax = 300, ymin = 200, ymax = 300, umin = 500, umax = 800, vmin = 500, vmax = 800;

enum { top = 0x8, bottom = 0x4, right = 0x2, left = 0x1 };

void myinit()

{

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(-200, 1200, -200, 1200);

    glMatrixMode(GL\_MODELVIEW);

}

outcode Computeoutcode(float x, float y)

{

    outcode code = 0;

    if (y > ymax)

        code = code | top;

    else if (y < ymin)

        code = code | bottom;

    if (x > xmax)

        code = code | right;

    else if (x < xmin)

        code = code | left;

    return code;

}

void drawline(float xrmin, float yrmin, float xrmax, float yrmax, float x1, float y1, float x2, float y2)

{

    glColor3f(0, 0, 1);

    glBegin(GL\_LINE\_LOOP);

    glVertex2f(xrmin, yrmin);

    glVertex2f(xrmin, yrmax);

    glVertex2f(xrmax, yrmax);

    glVertex2f(xrmax, yrmin);

    glEnd();

    glColor3f(1, 0, 0);

    glBegin(GL\_LINES);

    glVertex2f(x1, y1);

    glVertex2f(x2, y2);

    glEnd();

    glFlush();

}

void cohensoutherland\_algo(float x1, float y1, float x2, float y2)

{

    bool accept = false, done = false;

    float x = 0, y = 0;

    outcode p1, p2, p = 0;

    p1 = Computeoutcode(x1, y1);

    p2 = Computeoutcode(x2, y2);

    float m = (y2 - y1) / (x2 - x1);

    do {

        if (!(p1 | p2))

        {

            accept = true;

            done = true;

        }

        else if (p1 & p2)

            done = true;

        else

        {

            p = p1 ? p1 : p2;

            if (p & bottom)

            {

                y = ymin;

                x = x1 + (ymin - y1) / m;

            }

            else if (p & top)

            {

                y = ymax;

                x = x1 + (ymax - y1) / m;

            }

            else if (p & right)

            {

                x = xmax;

                y = y1 + (xmax - x1) \* m;

            }

            else

            {

                x = xmin;

                y = y1 + (xmin - x1) \* m;

            }

            if (p == p2)

            {

                x2 = x;

                y2 = y;

                p2 = Computeoutcode(x2, y2);

            }

            if (p == p1)

            {

                x1 = x;

                y1 = y;

                p1 = Computeoutcode(x1, y1);

            }

        }

    } while (!done);

    if (accept)

    {

        float sx = (umax - umin) / (xmax - xmin);

        float sy = (vmax - vmin) / (ymax - ymin);

        x1 = sx \* x1 + umin - sx \* xmin;

        y1 = sy \* y1 + vmin - sy \* ymin;

        x2 = sx \* x2 + umin - sx \* xmin;

        y2 = sy \* y2 + vmin - sy \* ymin;

        drawline(umin, vmin, umax, vmax, x1, y1, x2, y2);

    }

}

void display()

{

    glClearColor(1, 1, 1, 1);

    glClear(GL\_COLOR\_BUFFER\_BIT);

    drawline(xmin, ymin, xmax, ymax, 180, 180, 320, 320);

    cohensoutherland\_algo(180, 180, 320, 320);

}

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

{

    glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

    glutInitWindowSize(500, 500);

    glutInitWindowPosition(0, 0);

    glutInit(&argc, argv);

    glutCreateWindow("Cohensoutherland Line Clipping");

    myinit();

    glutDisplayFunc(display);

    glutMainLoop();

}

## **Prog 6**

#include "stdafx.h"

#include<math.h>

#include<glut.h>

void obj(double tx,double ty,double tz,double sx,double sy,double sz)

{

glRotated(50,0,1,0);

glRotated(10,-1,0,0);

glRotated(11.7,0,0,-1);

glTranslated(tx,ty,tz);

glScaled(sx,sy,sz);

glutSolidCube(1);

glLoadIdentity();

}

void display()

{

glViewport(0,0,700,700);

glClear(GL\_COLOR\_BUFFER\_BIT|GL\_DEPTH\_BUFFER\_BIT);

obj(0,0,0.5,1,1,0.04); // three walls

obj(0,-0.5,0,1,0.04,1);

obj(-0.5,0,0,0.04,1,1);

obj(0,-0.3,0,0.02,0.2,0.02); // four table legs

obj(0,-0.3,-0.4,0.02,0.2,0.02);

obj(0.4,-0.3,0,0.02,0.2,0.02);

obj(0.4,-0.3,-0.4,0.02,0.2,0.02);

obj(0.2,-0.18,-0.2,0.6,0.02,0.6); // table top

glRotated(50,0,1,0);

glRotated(10,-1,0,0);

glRotated(11.7,0,0,-1);

glTranslated(0.3,-0.1,-0.3);

glutSolidTeapot(0.09);

glFlush();

glLoadIdentity();

}

void main()

{

float blue[]={1,1,1,1};

float light\_pos[]={27,80,2,3};

glutInitWindowSize(700,700);

glutCreateWindow("scene");

glutDisplayFunc(display);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glMaterialfv(GL\_FRONT,GL\_AMBIENT,blue);

glLightfv(GL\_LIGHT0,GL\_POSITION,light\_pos);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

}

## **Prog 7**

#include "stdafx.h"

#include <stdio.h>

#include<glut.h>

typedef float point[3];

point v[]={{0.0, 0.0, 1.0},{0.0, 0.9, -0.3}, {-0.8, -0.5, -0.3}, {0.8, -0.5, -0.3}};

int n;

void triangle( point a, point b, point c)

{

glBegin(GL\_POLYGON);

glNormal3fv(a);

glVertex3fv(a);

glVertex3fv(b);

glVertex3fv(c);

glEnd();

}

void divide\_triangle(point a, point b, point c, int m)

{

point v1, v2, v3;

int j;

if(m>0)

{

for(j=0; j<3; j++) v1[j]=(a[j]+b[j])/2;

for(j=0; j<3; j++) v2[j]=(a[j]+c[j])/2;

for(j=0; j<3; j++) v3[j]=(b[j]+c[j])/2;

divide\_triangle(a, v1, v2, m-1);

divide\_triangle(c, v2, v3, m-1);

divide\_triangle(b, v3, v1, m-1);

}

else triangle(a,b,c); // draw triangle at end of recursion

}

void tetrahedron( int m)

{

glColor3f(1.0,0.0,0.0);

divide\_triangle(v[0], v[1], v[2], m);

glColor3f(0.0,0.0,0.0);

divide\_triangle(v[3], v[2], v[1], m);

glColor3f(0.0,0.0,1.0);

divide\_triangle(v[0], v[3], v[1], m);

glColor3f(0.0,1.0,0.0);

divide\_triangle(v[0], v[2], v[3], m);

}

void display(void)

{

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

tetrahedron(n);

glFlush();

}

void myReshape(int w, int h)

{

glClearColor (1.0, 1.0, 1.0, 1.0);

glViewport(0, 0, w, h);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

if (w <= h)

glOrtho(- 2.0, 2.0, -2.0 \* (GLfloat) h / (GLfloat) w, 2.0 \* (GLfloat) h / (GLfloat) w, -10.0, 10.0);

else

    glOrtho(-2.0 \* (GLfloat) w / (GLfloat) h,2.0 \* (GLfloat) w / (GLfloat) h, -2.0, 2.0, -10.0, 10.0);

glMatrixMode(GL\_MODELVIEW);

glutPostRedisplay();

}

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

{

printf(" No. of Divisions ? ");

scanf\_s("%d",&n); glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB |

GLUT\_DEPTH); glutInitWindowSize(500, 500);

glutCreateWindow("3D Gasket");

glutReshapeFunc(myReshape);

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

return 1;

}

## **Prog 8**

#include"stdafx.h"

#include<glut.h>

#include<stdio.h>

#include<math.h>

#define PI 3.1416

typedef struct point

{

GLfloat x, y, z;

};

void bino(int n, int \*C)

{

int k, j;

for(k=0;k<=n;k++)

{

C[k]=1;

for(j=n;j>=k+1; j--)

C[k]\*=j;

for(j=n-k;j>=2;j--)

C[k]/=j;

}

}

void computeBezPt(float u, point \*pt1, int cPt, point \*pt2, int \*C)

{

int k, n=cPt-1;

float bFcn;

pt1 ->x =pt1 ->y = pt1->z=0.0;

for(k=0; k< cPt; k++)

{

bFcn = C[k] \* pow(u, k) \* pow( 1-u, n-k);

pt1 ->x += pt2[k].x \* bFcn;

pt1 ->y += pt2[k].y \* bFcn;

pt1 ->z += pt2[k].z \* bFcn;

}

}

void bezier(point \*pt1, int cPt, int bPt)

{

point bcPt;

float u;

int \*C, k;

C= new int[cPt];

bino(cPt-1, C);

glBegin(GL\_LINE\_STRIP);

for(k=0; k<=bPt; k++)

    {

u=float(k)/float(bPt);

computeBezPt(u, &bcPt, cPt, pt1, C);

glVertex2f(bcPt.x, bcPt.y);

}

glEnd();

delete[]C;

}

float theta = 0;

void display()

{

glClear(GL\_COLOR\_BUFFER\_BIT);

int nCtrlPts = 4, nBCPts =20;

point ctrlPts[4] = {{100, 400, 0}, {150, 450, 0}, {250, 350, 0},

{300, 400, 0}};

ctrlPts[1].x +=50\*sin(theta \* PI/180.0);

ctrlPts[1].y +=25\*sin(theta \* PI/180.0);

ctrlPts[2].x -= 50\*sin((theta+30) \* PI/180.0);

ctrlPts[2].y -= 50\*sin((theta+30) \* PI/180.0);

ctrlPts[3].x -= 25\*sin((theta) \* PI/180.0);

ctrlPts[3].y += sin((theta-30) \* PI/180.0);

theta+=0.2;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0, 1.0, 1.0);

glPointSize(5);

glPushMatrix();

glLineWidth(5);

glColor3f(1, 0.4, 0.2); //Indian flag: Orange color code

for(int i=0;i<50;i++)

{

glTranslatef(0, -0.8, 0);

bezier(ctrlPts, nCtrlPts, nBCPts);

}

glColor3f(1, 1, 1); //Indian flag: white color code

for(int i=0;i<50;i++)

{

glTranslatef(0, -0.8, 0);

bezier(ctrlPts, nCtrlPts, nBCPts);

}

glColor3f(0, 1, 0); //Indian flag: green color code

for(int i=0;i<50;i++)

{

glTranslatef(0, -0.8, 0);

bezier(ctrlPts, nCtrlPts, nBCPts);

}

glPopMatrix();

glColor3f(0.7, 0.5,0.3);

glLineWidth(5);

glBegin(GL\_LINES);

glVertex2f(100,400);

glVertex2f(100,40);

glEnd();

glutPostRedisplay();

glutSwapBuffers();

}

void init()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

}

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

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB);

glutInitWindowPosition(0, 0);

glutInitWindowSize(500,500);

glutCreateWindow("Bezier Curve");

init();

glutDisplayFunc(display);

glutMainLoop();

}

## **Prog 9**

#include "stdafx.h"

#include <stdlib.h>

#include <stdio.h>

#include <glut.h>

float x1,x2,x3,x4,y1,y2,y3,y4;

int fillFlag=0;

void edgedetect(float x1,float y1,float x2,float y2,int \*le,int \*re)  {

    float mx,x,temp; int i;

    if((y2-y1)<0) {

        temp=y1;y1=y2;y2=temp;

        temp=x1;x1=x2;x2=temp;

    }

    if((y2-y1)!=0)

        mx=(x2-x1)/(y2-y1);

    else

        mx=x2-x1;

    x=x1;

    for(i=y1;i<=y2;i++)   {

        if(x<(float)le[i])

            le[i]=(int)x;

        if(x>(float)re[i])

            re[i]=(int)x;

        x+=mx;

    } }

void draw\_pixel(int x,int y)

{

    glColor3f(1.0,1.0,0.0);

    glBegin(GL\_POINTS);

    glVertex2i(x,y);

    glEnd();

}  void scanfill(float x1,float y1,float x2,float y2,float x3,float y3,float x4,float y4) {

    int le[500],re[500];

    int i,y;

    for(i=0;i<500;i++)  {

        le[i]=500;

        re[i]=0;

    }

    edgedetect(x1,y1,x2,y2,le,re);

    edgedetect(x2,y2,x3,y3,le,re);

    edgedetect(x3,y3,x4,y4,le,re);

    edgedetect(x4,y4,x1,y1,le,re);

    for(y=0;y<500;y++)   {

        for(i=(int)le[y];i<(int)re[y];i++)

            draw\_pixel(i,y);

    }

}

void display() {

    x1=200.0;y1=200.0;x2=100.0;y2=300.0;x3=200.0;y3=400.0;x4=300.0;y4=300.0;

    glClear(GL\_COLOR\_BUFFER\_BIT);

    glColor3f(0.0, 0.0, 1.0);

    glBegin(GL\_LINE\_LOOP);

    glVertex2f(x1,y1);

    glVertex2f(x2,y2);

    glVertex2f(x3,y3);

    glVertex2f(x4,y4);

    glEnd();

    if(fillFlag==1)

        scanfill(x1,y1,x2,y2,x3,y3,x4,y4);

    glFlush();

}    void init()  {

    glClearColor(0.0,0.0,0.0,1.0);

    glColor3f(1.0,0.0,0.0);

    glPointSize(1.0);

    glMatrixMode(GL\_PROJECTION);

    glLoadIdentity();

    gluOrtho2D(0.0,499.0,0.0,499.0);

}

void fillMenu(int option) {

    if(option==1)

        fillFlag=1;

    if(option==2)

        fillFlag=2;

    display();

}

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

    glutInit(&argc,argv);

    glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB);

    glutInitWindowSize(500,500);

    glutInitWindowPosition(0,0);

    glutCreateWindow("Filling a Polygon using Scan-line Algorithm");

    init();

    glutDisplayFunc(display);

    glutCreateMenu(fillMenu);

    glutAddMenuEntry("Fill Polygon",1);

    glutAddMenuEntry("Empty Polygon",2);

    glutAttachMenu(GLUT\_RIGHT\_BUTTON);

    glutMainLoop();

}