**Scanfill**

# include <stdio.h>

# include <GL/glut.h>

int xa,ya, xb, yb, xc, yc, xd, yd ;

int flag=0;

void edgedetect(int xa,int ya,int xb,int yb,float le[],float re[]){

float mx, x;

int i;

if((yb-ya)<0){

int temp;

temp=xb;

xb=xa;

xa=temp;

temp=yb;

yb=ya;

ya=temp;

}

if((yb-ya)!=0)

mx= (xb-xa)/(float)(yb-ya);

else

mx=xb-xa;

x=xa;

for(i=ya;i<yb;i++){

if(x<(float)le[i])

le[i]=(int)x;

if(x>(float)re[i])

re[i]=(int)x;

x+=mx;

}

}

void scanfill(int xa,int ya,int xb,int yb,int xc,int yc,int xd,int yd){

int y, i;

float le[500], re[500];

for(i=0;i<500;i++){

le[i]=500;

re[i]=0;

}

edgedetect(xa,ya,xb,yb,le,re);

edgedetect(xb,yb,xc,yc,le,re);

edgedetect(xd,yd,xc,yc,le,re);

edgedetect(xd,yd,xa,ya,le,re);

for(y=0;y<500;y++){

if(le[y]<=re[y]){

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

glColor3f(1,0,0);

glBegin(GL\_POINTS);

glVertex2f(i,y);

glEnd();

}

}

}

}

void init(){

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,500,0,500);

glMatrixMode(GL\_MODELVIEW);

}

void display(){

glClearColor(1,1,1,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(xa,ya);

glVertex2f(xb,yb);

glVertex2f(xc,yc);

glVertex2f(xd,yd);

glEnd();

if(flag==1)

scanfill(xa,ya,xb,yb,xc,yc,xd,yd);

glFlush();

}

void fillmenu(int option){

if(option==1)

flag=1;

if(option==2)

flag=0;

display();

}

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

printf("enter the 4 point\n");

scanf("%d%d%d%d%d%d%d%d",&xa,&ya,&xb,&yb,&xc,&yc,&xd,&yd);

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("CG PROGRAM");

glutInitWindowPosition(0,0);

glutInitWindowSize(500,500);

init();

glutDisplayFunc(display);

glutCreateMenu(fillmenu);

glutAddMenuEntry("Fill polygon!!",1);

glutAddMenuEntry("Empty polygon!!",2);

glutAttachMenu(GLUT\_LEFT\_BUTTON);

glutMainLoop();

}

**Gasket**

#include<stdio.h>

#include<GL/glut.h>

float v[4][3]={{0,0,1},{0,1,-1},{-1,-1,-1},{1,-1,-1}};

int m;

void triangle(float a[3],float b[3],float c[3])

{

glBegin(GL\_TRIANGLES);

glVertex3fv(a);

glVertex3fv(b);

glVertex3fv(c);

glEnd();

}

void divide\_triangle(float a[3],float b[3],float c[3],int m)

{

float v1[3],v2[3],v3[3];

if(m>0)

{

for(int j=0;j<3;j++)

{

v1[j]=(a[j]+b[j])/2;

v2[j]=(b[j]+c[j])/2;

v3[j]=(a[j]+c[j])/2;

}

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

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

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

}

else

triangle(a,b,c);

}

void init()

{

glMatrixMode(GL\_PROJECTION);

glOrtho(-2,2,-2,2,-2,2);

glMatrixMode(GL\_MODELVIEW);

}

void display()

{

glClearColor(1,1,1,1);

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

glColor3f(1,0,0);

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

glColor3f(0,1,0);

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

glColor3f(0,0,1);

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

glColor3f(1,0,1);

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

glFlush();

}

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

{

printf("enter no. of divisions");

scanf("%d",&m);

glutInit(&argc,argv);

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

glutCreateWindow("Gasket");

glutInitWindowPosition(0,0);

glutInitWindowSize(500,500);

init();

glutDisplayFunc(display);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

}

**Clip**

#include<stdio.h>

#include<GL/glut.h>

int xmin=50,xmax=100,ymin=50,ymax=100;

int x0,y0,x1,y1;

const int bottom=1;

const int left=2;

const int top=4;

const int right=8;

void init()

{

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,500,0,500);

glMatrixMode(GL\_MODELVIEW);

}

int compute(int x,int y)

{

int code=0;

if(x<xmin)

code|=left;

if(x>xmax)

code|=right;

if(y<ymin)

code|=bottom;

if(y>ymax)

code|=top;

return code;

}

void cohen(int x0,int y0,int x1,int y1)

{

int x,y;

float m=(y1-y0)/(x1-x0);

bool done=false,accept=false;

int code0=compute(x0,y0);

int code1=compute(x1,y1);

do

{

if(code0 & code1)

{

done=true;

}

else if(!(code0|code1))

{

done=true;

accept=true;

}

else

{

int out=code0?code0:code1;

if(out & bottom)

{

x=x0+((ymin-y0)/m);

y=ymin;

}

if(out & top)

{

x=x0+((ymax-y0)/m);

y=ymax;

}

if(out & left)

{

x=xmin;

y=y0+((xmin-x0)\*m);

}

if(out & right)

{

x=xmax;

y=y0+((xmin-x0)\*m);

}

if(out==code0)

{

x0=x;

y0=y;

code0=compute(x0,y0);

}

if(out==code1)

{

x1=x;

y1=y;

code1=compute(x1,y1);

}

}

}while(!done);

if(accept==true)

{

glTranslatef(100,100,0);

glColor3f(0,1,0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(xmin,ymin);

glVertex2f(xmax,ymin);

glVertex2f(xmax,ymax);

glVertex2f(xmin,ymax);

glEnd();

glBegin(GL\_LINES);

glVertex2f(x0,y0);

glVertex2f(x1,y1);

glEnd();

}

}

void display()

{

glClearColor(1,1,1,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

glBegin(GL\_LINE\_LOOP);

glVertex2f(xmin,ymin);

glVertex2f(xmax,ymin);

glVertex2f(xmax,ymax);

glVertex2f(xmin,ymax);

glEnd();

glBegin(GL\_LINES);

glVertex2f(x0,y0);

glVertex2f(x1,y1);

glEnd();

cohen(x0,y0,x1,y1);

glFlush();

}

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

{

printf("enter x0,y0,x1 and y1 value");

scanf("%d\t%d\t%d\t%d",&x0,&y0,&x1,&y1);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("Clippingprog");

glutInitWindowPosition(0,0);

glutInitWindowSize(500,500);

init();

glutDisplayFunc(display);

glutMainLoop();

}

**line**

#include<stdio.h>

#include<GL/glut.h>

float m,x0,x1,y0,y1;

void init()

{

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,100,0,100);

glMatrixMode(GL\_MODELVIEW);

}

void draw\_pixel(float x,float y)

{

printf("%f %f\n",x,y);

glPointSize(3.0);

glColor3f(1,0,0);

glBegin(GL\_POINTS);

glVertex2f(x,y);

glEnd();

}

void linedrawing1(float x0,float y0,float x1,float y1,float m)

{

float x,y,dx,dy,pneg,ppos,p;

if(x0>x1)

{

float tmp;

tmp=x0;

x0=x1;

x1=tmp;

tmp=y0;

y0=y1;

y1=tmp;

}

dx=abs(x1-x0);

dy=abs(y1-y0);

x=x0;

y=x0;

pneg=2\*dy;

ppos=(2\*dy)-(2\*dx);

p=(2\*dy)-dx;

draw\_pixel(x,y);

while(x<x1)

{

x++;

if(p<0)

{

p=p+pneg;

}

else

{

if(m<0)

y--;

else

{

y++;

p=p+ppos;

}

}

draw\_pixel(x,y);

}

}

void linedrawing2(float x0,float y0,float x1,float y1,float m)

{

float x,y,dx,dy,pneg,ppos,p;

if(y0>y1)

{

float tmp;

tmp=x0;

x0=x1;

x1=tmp;

tmp=y0;

y0=y1;

y1=tmp;

}

dx=abs(x1-x0);

dy=abs(y1-y0);

x=x0;

y=y0;

pneg=2\*dx;

ppos=(2\*dx)-(2\*dy);

p=(2\*dx)-dy;

draw\_pixel(x,y);

while(y<y1)

{

y++;

if(p<0)

{

p=p+pneg;

}

else

{

x++;

p=p+ppos;

}

draw\_pixel(x,y);

}

}

void display()

{

glClearColor(1,1,1,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1,0,0);

m=(y1-y0)/(x1-x0);

if(m<1)

linedrawing1(x0,y0,x1,y1,m);

else

linedrawing2(x0,y0,x1,y1,m);

glFlush();

}

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

{

printf("enter x0,y0 and x1,y1 values");

scanf("%f\t%f\t%f\t%f",&x0,&y0,&x1,&y1);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("lineprog");

glutInitWindowPosition(0,0);

glutInitWindowSize(500,500);

init();

glutDisplayFunc(display);

glutMainLoop();

}

**tri.c**

#include<stdio.h>

#include<GL/glut.h>

float rot\_mat[3][3]={{0},{0},{0}};

float result[3][3]={{0},{0},{0}};

float triangle[3][3]={{100,150,200},{100,200,100},{1,1,1}};

float theta=0;

float h=200,k=100;

float m=0,n=0;

int flag;

void init()

{

glMatrixMode(GL\_PROJECTION);

gluOrtho2D(0,500,0,500);

glMatrixMode(GL\_MODELVIEW);

}

void drawtriangle(float a[3][3])

{

glBegin(GL\_POLYGON);

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

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

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

glEnd();

}

void multiply()

{

int i,j,p;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

result[i][j]=0;

for(p=0;p<3;p++)

{

result[i][j]+=rot\_mat[i][p]\*triangle[p][j];

}

}

}

}

void rotate(float m,float n)

{

rot\_mat[0][0]=cos(theta);

rot\_mat[0][1]=-sin(theta);

rot\_mat[0][2]=m;

rot\_mat[1][0]=sin(theta);

rot\_mat[1][1]=cos(theta);

rot\_mat[1][2]=n;

rot\_mat[2][0]=0;

rot\_mat[2][1]=0;

rot\_mat[2][2]=1;

multiply();

}

void display()

{

glClearColor(1,1,1,1);

glClear(GL\_COLOR\_BUFFER\_BIT);

if(flag==0)

{

glColor3f(1,0,0);

drawtriangle(triangle);

}

if(flag==1)

{

glColor3f(0,1,0);

rotate(0,0);

drawtriangle(result);

}

if(flag==2)

{

glColor3f(0,0,1);

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

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

rotate(m,n);

drawtriangle(result);

}

glFlush();

}

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

{

printf("enter theta value");

scanf("%f",&theta);

theta=theta\*(3.14/180);

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutCreateWindow("Triangleprog");

glutInitWindowPosition(0,0);

glutInitWindowSize(500,500);

init();

glutDisplayFunc(display);

glutMainLoop();

}

**Cube3d**

#include<stdio.h>

#include<GL/glut.h>

void init();

void display();

float v[8][3]={{-1,-1,-1},{-1,1,-1},{1,1,-1},{1,-1,-1},{-1,-1,1},{-1,1,1},{1,1,1},{1,-1,1}};

int theta[3]={0,0,0};

int axis=0;

void init()

{

glMatrixMode(GL\_PROJECTION);

glOrtho(-2,2,-2,2,-2,2);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glBegin(GL\_POLYGON);

glVertex3fv(v[a]);

glVertex3fv(v[b]);

glVertex3fv(v[c]);

glVertex3fv(v[d]);

glEnd();

}

void cube()

{

glColor3f(1,0,0);

polygon(0,1,2,3);

glColor3f(0,1,0);

polygon(3,2,6,7);

glColor3f(0,0,1);

polygon(0,1,5,4);

glColor3f(1,1,0);

polygon(4,5,6,7);

glColor3f(1,0,1);

polygon(1,2,6,5);

glColor3f(0,1,1);

polygon(0,3,7,4);

glColor3f(1,0,0);

}

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

{

if(button==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN)

axis=0;

if(button==GLUT\_RIGHT\_BUTTON && state==GLUT\_DOWN)

axis=1;

if(button==GLUT\_MIDDLE\_BUTTON && state==GLUT\_DOWN)

axis=2;

}

void spin()

{

theta[axis]+=1.0;

if(theta[axis]>360)

theta[axis]=0;

display();

}

void display()

{

glClearColor(1,1,1,1);

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

glLoadIdentity();

glRotatef(theta[0],1,0,0);

glRotatef(theta[1],0,1,0);

glRotatef(theta[2],0,0,1);

cube();

glFlush();

glutSwapBuffers();

}

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

{

glutInit(&argc,argv);

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

glutCreateWindow("gg");

glutInitWindowPosition(100,100);

glutInitWindowSize(1500,1500);

init();

glutDisplayFunc(display);

glutIdleFunc(spin);

glutMouseFunc(mouse);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

}

**Cubecamera**

#include<stdio.h>

#include<GL/glut.h>

float v[8][3]={{-1,-1,-1},{-1,1,-1},{1,1,-1},{1,-1,-1},{-1,-1,1},{-1,1,1},{1,1,1},{1,-1,1}};

int theta[3]={0,0,0};

int axis=0;

float viewer[3]={0,0,5};

void init()

{

glMatrixMode(GL\_PROJECTION);

//glOrtho(-2,2,-2,2,2,20);

glFrustum(-2,2,-2,2,2,10);

glMatrixMode(GL\_MODELVIEW);

}

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

{

glBegin(GL\_POLYGON);

glVertex3fv(v[a]);

glVertex3fv(v[b]);

glVertex3fv(v[c]);

glVertex3fv(v[d]);

glEnd();

}

void cube()

{

glColor3f(1,0,0);

polygon(0,1,2,3);

glColor3f(0,1,0);

polygon(3,2,6,7);

glColor3f(0,0,1);

polygon(0,1,5,4);

glColor3f(1,1,0);

polygon(4,5,6,7);

glColor3f(0,1,1);

polygon(1,2,6,5);

glColor3f(0,0,0);

polygon(0,3,7,4);

}

void display()

{

glClearColor(1,1,1,1);

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

glLoadIdentity();

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

glRotatef(theta[0],1,0,0);

glRotatef(theta[1],0,1,0);

glRotatef(theta[2],0,0,1);

cube();

glFlush();

glutSwapBuffers();

}

void spin()

{

theta[axis]+=1.0;

if(theta[axis]>360)

theta[axis]=0;

display();

}

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

{

if(button==GLUT\_LEFT\_BUTTON && state==GLUT\_DOWN)

axis=0;

if(button==GLUT\_RIGHT\_BUTTON && state==GLUT\_DOWN)

axis=1;

if(button==GLUT\_MIDDLE\_BUTTON && state==GLUT\_DOWN)

axis=2;

spin();

}

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

{

if(key=='x')

viewer[0]-=1;//camera value is decrementing so the object zooms in

if(key=='X') //object doesn’t rotate. The camera rotates according to the position

viewer[0]+=1; //given

if(key=='y')

viewer[1]-=1;

if(key=='Y')

viewer[1]+=1;

if(key=='z')

viewer[2]-=1;

if(key=='Z')

viewer[2]+=1;

display();

}

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

{

glutInit(&argc,argv);

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

glutCreateWindow("CUBE3D");

glutInitWindowPosition(0,0);

glutInitWindowSize(500,500);

init();

glutDisplayFunc(display);

//glutIdleFunc(spin);

glutMouseFunc(mouse);

glutKeyboardFunc(keys);

glEnable(GL\_DEPTH\_TEST);

glutMainLoop();

}

**Teapot**

**#include<GL/glut.h>**

**void wall(double thickness)**

**{**

**glPushMatrix();**

**glTranslated(0.5,0.5\*thickness,0.5);**

**glScaled(1.0,thickness,1.0);**

**glutSolidCube(1.0);**

**glPopMatrix();**

**}**

**void tableleg(double thick,double len)**

**{**

**glPushMatrix();**

**glTranslated(0,len/2,0);**

**glScaled(thick,len,thick);**

**glutSolidCube(1.0);**

**glPopMatrix();**

**}**

**void table(double topwid,double topthick,double legthick,double leglen)**

**{**

**glPushMatrix();**

**glTranslated(0,leglen,0);**

**glScaled(topwid,topthick,topwid);**

**glutSolidCube(1.0);**

**glPopMatrix();**

**double dist=0.95\*topwid/2.0-legthick/2.0;**

**glPushMatrix();**

**glTranslated(dist,0,dist);**

**tableleg(legthick,leglen);**

**glTranslated(0,0,-2\*dist);**

**tableleg(legthick,leglen);**

**glTranslated(-2\*dist,0,2\*dist);**

**tableleg(legthick,leglen);**

**glTranslated(0,0,-2\*dist);**

**tableleg(legthick,leglen);**

**glPopMatrix();**

**}**

**void display()**

**{**

**glClearColor(1,1,1,1);**

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

**GLfloat mat\_amb[]={1.111f,0.1f,0.7f,1.0f};**

**GLfloat mat\_diff[]={0.7f,0.7f,0.7f,1.0f};**

**GLfloat mat\_spec[]={0.7f,0.7f,0.7f,1.0f};**

**GLfloat mat\_shine[]={50.0f};**

**glMaterialfv(GL\_FRONT,GL\_AMBIENT,mat\_amb);**

**glMaterialfv(GL\_FRONT,GL\_DIFFUSE,mat\_diff);**

**glMaterialfv(GL\_FRONT,GL\_SPECULAR,mat\_spec);**

**glMaterialfv(GL\_FRONT,GL\_SHININESS,mat\_shine);**

**GLfloat lightintensity[]={0.7f,0.7f,0.7f,1.0f};**

**GLfloat lightposition[]={2.0f,6.0f,3.0f,0.0f};**

**glLightfv(GL\_LIGHT0,GL\_POSITION,lightposition);**

**glLightfv(GL\_LIGHT0,GL\_DIFFUSE,lightintensity);**

**glMatrixMode(GL\_PROJECTION);**

**glLoadIdentity();**

**glOrtho(-1,1,-1,1,0.1,100.0);**

**glMatrixMode(GL\_MODELVIEW);**

**gluLookAt(2.3,1.3,2.0,0.0,0.25,0.0,0.0,0.1,0.0);**

**glPushMatrix();**

**glTranslated(0.4,0.4,0.37);**

**glutSolidTeapot(0.089);**

**glPopMatrix();**

**glPushMatrix();**

**glTranslated(0.4,0,0.4);**

**table(0.6,0.07,0.06,0.3);**

**glPopMatrix();**

**glPushMatrix();**

**glRotated(90,0,0,1);**

**wall(0.02);**

**glPopMatrix();**

**glPushMatrix();**

**glRotated(-90,1,0,0);**

**wall(0.02);**

**glPopMatrix();**

**glPushMatrix();**

**glRotated(0,0,1,0);**

**wall(0.02);**

**glPopMatrix();**

**glFlush();**

**}**

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

**{**

**glutInit(&argc,argv);**

**glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE|GLUT\_DEPTH);**

**glutInitWindowSize(640,480);**

**glutCreateWindow("TEA POT ON the table");**

**glutDisplayFunc(display);**

**glEnable(GL\_LIGHTING);**

**glEnable(GL\_LIGHT0);**

**glShadeModel(GL\_SMOOTH);**

**glEnable(GL\_NORMALIZE);**

**glEnable(GL\_DEPTH\_TEST);**

**glViewport(0,0,640,480);**

**glutMainLoop();**

**}**

**Bezier Curve**

#include<GL/glut.h>

#include<stdio.h>

#include<math.h>

#define PI 3.1416

GLsizei winWidth=600,winHeight=600;

GLfloat xwcMin=0.0,xwcMax=130.0;

GLfloat ywcMin=0.0,ywcMax=130.0;

int flag=0;

struct wcPt3D

{

GLfloat x,y,z;

};

void bino(GLint n,GLint \*C)

{

GLint 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(GLfloat u,wcPt3D \*bezPt,GLint nCtrlPts,wcPt3D \*ctrlPts,GLint \*C)

{

GLint k,n=nCtrlPts-1;

GLfloat bezBlendFcn;

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

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

{

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

bezPt->x+=ctrlPts[k].x\*bezBlendFcn;

bezPt->y+=ctrlPts[k].y\*bezBlendFcn;

bezPt->z+=ctrlPts[k].z\*bezBlendFcn;

}

}

void bezier(wcPt3D \*ctrlPts,GLint nCtrlPts,GLint nBezCurvePts)

{

wcPt3D bexCurvePt;

GLfloat u;

GLint \*C,k;

C=new GLint[nCtrlPts];

bino(nCtrlPts-1,C);

glBegin(GL\_LINE\_STRIP);

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

{

u=GLfloat(k)/GLfloat(nBezCurvePts);

computeBezPt(u,&bexCurvePt,nCtrlPts,ctrlPts,C);

glVertex2f(bexCurvePt.x,bexCurvePt.y);

}

glEnd();

delete[]C;

}

void displayFcn()

{

GLint nCtrlPts=4,nBezCurvePts=20;

static float theta=0;

wcPt3D ctrlPts[4]={

{20,100,0},

{30,110,0},

{50,90,0},

{60,100,0}};

ctrlPts[1].x+=10\*sin(theta\*PI/180.0);

ctrlPts[1].y+=5\*sin(theta\*PI/180.0);

ctrlPts[2].x-=10\*sin((theta+30)\*PI/180.0);

ctrlPts[2].y-=10\*sin((theta+30)\*PI/180.0);

ctrlPts[3].x-=4\*sin((theta)\*PI/180.0);

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

if(flag==1)

theta+=2;

theta+=0;

glClear(GL\_COLOR\_BUFFER\_BIT);

glColor3f(1.0,1.0,1.0);

glPointSize(5);

glPushMatrix();

glLineWidth(5);

glColor3f(255/255,153/255.0,51/255.0);

for(int i=0;i<8;i++)

{

glTranslatef(0,-0.8,0);

bezier(ctrlPts,nCtrlPts,nBezCurvePts);

}

glColor3f(1,1,1);

for(int i=0;i<8;i++)

{

glTranslatef(0,-0.8,0);

bezier(ctrlPts,nCtrlPts,nBezCurvePts);

}

glColor3f(19/255.0,136/255.0,8/255.0);

for(int i=0;i<8;i++)

{

glTranslatef(0,-0.8,0);

bezier(ctrlPts,nCtrlPts,nBezCurvePts);

}

glPopMatrix();

glColor3f(0.7,0.5,0.3);

glLineWidth(5);

glBegin(GL\_LINES);

glVertex2f(20,100);

glVertex2f(20,40);

glEnd();

glFlush();

glutPostRedisplay();

glutSwapBuffers();

}

void winReshapeFun(GLint newWidth,GLint newHeight)

{

glViewport(0,0,newWidth,newHeight);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(xwcMin,xwcMax,ywcMin,ywcMax);

glClear(GL\_COLOR\_BUFFER\_BIT);

}

void fillmenu(int option)

{

if(option==1)

flag=1;

if(option==2)

flag=0;

displayFcn();

}

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

{

glutInit(&argc,argv);

glutInitDisplayMode(GLUT\_RGB|GLUT\_DOUBLE);

glutInitWindowPosition(50,50);

glutInitWindowSize(winWidth,winHeight);

glutCreateWindow("BEZIER CURVE");

glutDisplayFunc(displayFcn);

glutReshapeFunc(winReshapeFun);

glutCreateMenu(fillmenu);

glutAddMenuEntry("Animation",1);

glutAddMenuEntry("Flag",2);

glutAttachMenu(GLUT\_LEFT\_BUTTON);

glutMainLoop();

}