```c
#include<GL/glut.h>
#include<stdio.h>

#define true 1;
#define false 0;
#define bool int;


double x,y;
int xmin=50,xmax=100,ymin=50,ymax=100;
const int RIGHT=8,LEFT=2,TOP=4,BOTTOM=1;
int outcode0,outcode1,outcodeout,done,accept;

int computeoutcode(double x,double y)
{
  int code=0;
  if(y>ymax)
  code|=TOP;
  else if(y<ymin)
  code|=BOTTOM;
  if(x>xmax)
  code|=RIGHT;
  else if(x<xmin)
  code|=LEFT;
  return code;
}


void LineClip(double x0,double y0,double x1,double y1)
{
  int accept=false;
  int done=false;
  outcode0=computeoutcode(x0,y0);
  outcode1=computeoutcode(x1,y1);
  do{
    if(!(outcode0|outcode1))
    {
      accept=true;
      done=true;
    }
    else if(outcode0&outcode1)
    {
      done=true;
    }
    else
    {
      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)
  {
    glPushMatrix();

    glTranslatef(100,100,0);
    glColor3f(1.0,0.0,0.0);
    glBegin(GL_LINE_LOOP);
    glVertex2i(50,50);
    glVertex2i(100,50);
    glVertex2i(100,100);
    glVertex2i(50,100);
    glEnd();

    glColor3f(1.0,0.0,1.0);
    glBegin(GL_LINES);
    glVertex2i(x0,y0);
    glVertex2i(x1,y1);
    glEnd();
    glPopMatrix();
    glFlush();
  }
}

void display()
{

  glClearColor(1,1,1,1);
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0,0.0,0.0);
  glBegin(GL_LINE_LOOP);
  glVertex2i(50,50);
  glVertex2i(100,50);
  glVertex2i(100,100);
  glVertex2i(50,100);
  glEnd();
  glColor3f(1.0,0.0,1.0);
  glBegin(GL_LINES);
  glVertex2i(60,20);
  glVertex2i(80,120);
  glVertex2i(80,20);
  glVertex2i(60,120);
  glEnd();
  LineClip(60,20,80,120);
  LineClip(80,20,60,120);

  glFlush();
}

void myInit()
{
  glMatrixMode(GL_PROJECTION);
  gluOrtho2D(0,300,0,300);
  glMatrixMode(GL_MODELVIEW);
}
```

```c
void main(int argc,char** argv)
{
  glutInit(&argc,argv);
  glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
  glutInitWindowPosition(0,0);
  glutInitWindowSize(1000,1000);
  glutCreateWindow("clipping");
  myInit();
  glutDisplayFunc(display);
  glutMainLoop();
}
```

```c
#include<GL/gl.h>
#include<stdio.h>
#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()
{
        float mat_amb[]={1.0f,0.0f,1.0f,1.0f};
        float mat_diff[]={1.0f,1.0f,1.0f,1.0f};
        float mat_spec[]={1.0f,0.0f,1.0f,1.0f};
        float 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);

        float lightintensity[]={0.7f,0.7f,0.7f,1.0f};
        float lightposition[]={2.0f,6.0f,3.0f,0.0f};

        glLightfv(GL_LIGHT0,GL_POSITION,lightposition);
        glLightfv(GL_LIGHT0,GL_DIFFUSE,lightintensity);

        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);

        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-1.5,1.5,-1,1,0.1,100);

        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        gluLookAt(2,1,3,0,0.25,0,0,1,0);


        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(-90,1,0,0);
        wall(0.02);
        glPopMatrix();


        glPushMatrix();
        wall(0.02);
        glPopMatrix();


        glPushMatrix();
```

```
        glRotated(90,0,0,1);
        wall(0.02);
        glPopMatrix();
        glFlush();
}

int main(int argc,char** argv)
{
        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);

        glutInitWindowSize(700,700);
        glutInitWindowPosition(10,10);
        glutCreateWindow("tea");

        glutDisplayFunc(display);

        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glEnable(GL_NORMALIZE);
        glEnable(GL_DEPTH_TEST);

        glutMainLoop();
}
```

```
#include<stdio.h>
#include<GL/glut.h>

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

int m;

void init()
{
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        glOrtho(-2,2,-2,2,-5,5);
        glMatrixMode(GL_MODELVIEW);

}

void triangle(float *a,float *b,float *c)
{
        glBegin(GL_TRIANGLES);
        glVertex3fv(a);
        glVertex3fv(b);
        glVertex3fv(c);
        glEnd();
}

void dividetriangle(float *a,float *b,float *c,int m)
{
```

```
        int i;
        float v1[3],v2[3],v3[3];
        if(m>0)
        {
                for(i=0;i<3;i++)
                {
                        v1[i]=(a[i]+b[i])/2;
                        v2[i]=(b[i]+c[i])/2;
                        v3[i]=(c[i]+a[i])/2;
                }
                dividetriangle(a,v1,v3,m-1);
                dividetriangle(b,v1,v2,m-1);
                dividetriangle(c,v2,v3,m-1);
        }
        else
                triangle(a,b,c);
}

void tetrahedron(int m)
{
        glColor3f(1,0,0);
        dividetriangle(v[0],v[1],v[2],m);
        glColor3f(0,0,1);
        dividetriangle(v[1],v[0],v[3],m);
        glColor3f(0,1,0);
        dividetriangle(v[0],v[2],v[3],m);
        glColor3f(1,1,0);
        dividetriangle(v[1],v[2],v[3],m);

}

void display()
{
        glClearColor(0,0,0,1);
        glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
        glLoadIdentity();
        gluLookAt(2,2,2,1,1,1,0,1,0);

        tetrahedron(m);

        glFlush();
}


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

        glutInit(&argc,argv);
        glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB|GLUT_DEPTH);
        printf("Enter Level of Sierpinski Gasket: ");
        scanf("%d",&m);
        glutInitWindowPosition(0,0);
        glutInitWindowSize(600,600);
        glutCreateWindow("Sierpinski Gasket");
        init();
        glEnable(GL_DEPTH_TEST);
        glutDisplayFunc(display);
        glutMainLoop();
}
```

```
#include<GL/glut.h>
#include<bits/stdc++.h>
using namespace std;
#define PI 3.1416

typedef struct wcPt3D
{
        float 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, wcPt3D *bezPt, int nCtrlPts, wcPt3D *ctrlPts, int *C)
{
        int k, n=nCtrlPts-1;
        float 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, int nCtrlPts, int nBezCurvePts)
{
        wcPt3D bezCurvePt;
        float u;
        int *C, k;
        C= new int[nCtrlPts];
        bino(nCtrlPts-1, C);
        glBegin(GL_LINE_STRIP);
        for(k=0; k<=nBezCurvePts; k++)
        {
                u=float(k)/float(nBezCurvePts);
```

```
                computeBezPt(u, &bezCurvePt, nCtrlPts, ctrlPts, C);
                glVertex2f(bezCurvePt.x, bezCurvePt.y);
        }
        glEnd();
        delete[] C;
}

void displayFcn()
{
        int nCtrlPts = 4, nBezCurvePts =20;
        printf("display called\n");
        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);
        printf("%f\n", theta);
        theta+=0.8;

        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);
        glBegin(GL_LINES);
        glVertex2f(20,100);
        glVertex2f(20,40);
        glEnd();
        glFlush();
        glutPostRedisplay();
        glutSwapBuffers();
}
```

```c
void winReshapeFun(int w, int h)
{
        glViewport(0, 0, w, h);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0, 130, 0, 130);
        glClear(GL_COLOR_BUFFER_BIT);
}

int main(int argc, char **argv)
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
        glutInitWindowPosition(50, 50);
        glutInitWindowSize(600, 600);
        glutCreateWindow("Bezier Curve");
        glutDisplayFunc(displayFcn);
        glutReshapeFunc(winReshapeFun);
        glutMainLoop();
        return 0;
}
```
