Program 8: Sierpinski Gasket

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

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


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;
    if(m > 0){
        float v1[3], v2[3], v3[3];
        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,v2,v1,m-1);
        dividetriangle(c,v3,v2,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(2,2,2,1,1,1,1,1,0);
    tetrahedron(n);
    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",&n);
    glutInitWindowPosition(0,0);
    glutInitWindowSize(600,600);
    glutCreateWindow("Sierpinski Gasket");
    init();
    glEnable(GL_DEPTH_TEST);
    glutDisplayFunc(display);
    glutMainLoop();
}
```

Program 7: Table

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


void wall(double thickness){
    glPushMatrix();
    glTranslated(0.5, 0.5*thickness, 0.5);
    glScaled(1, thickness, 1);
    glutSolidCube(1.0);
    glPopMatrix();
}


void tableleg(double thickness, double len){
    glPushMatrix();
    glTranslated(0, len/2, 0);
    glScaled(thickness, len, thickness);
    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("Teapot");

    glutDisplayFunc(display);

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

    glutMainLoop();
}
```

Program 5: Line Clipping

```
#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)
{
  accept=false;
  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 init()
{
  glMatrixMode(GL_PROJECTION);
  gluOrtho2D(0,300,0,300);
  glMatrixMode(GL_MODELVIEW);
}



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

Program 4: Cube

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

float ver[][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}};
float theta[]={0,0,0};
int viewer[3] = {0,0,5};
int axis=2;

void polygon(int a,int b,int c,int d)
{
    glBegin(GL_POLYGON);
    glColor3f(.5,0,0.6);
    glVertex3fv(ver[a]);
    glColor3f(1,0.2,0.2);
    glVertex3fv(ver[b]);
```

```
    glColor3f(1,0.5,0.5);
    glVertex3fv(ver[c]);
    glColor3f(1,1,0);
    glVertex3fv(ver[d]);
    glEnd();
}



void color_cube()
{
    glColor3f(1.0,1.0,0.0);
    polygon(0,3,2,1);

    glColor3f(1.0,0.7,0.0);
    polygon(2,3,7,6);

    glColor3f(1.0,1.0,1.0);
    polygon(0,4,7,3);

    glColor3f(1.0,0.0,1.0);
    polygon(1,2,6,5);

    glColor3f(0.4,0.6,0.8);
    polygon(4,5,6,7);

    glColor3f(0.8,0.2,0.1);
    polygon(0,1,5,4);

}
void init()
{

    glMatrixMode(GL_PROJECTION);
    glFrustum(-2, 2, -2, 2, 2, 10);
    glMatrixMode(GL_MODELVIEW);

}

void display()
{
    glClearColor(0,0,0,1);

    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);
    color_cube();

    glFlush();
    glutSwapBuffers();

}
void spin_cube()
{
    theta[axis]+=1.0;
    if(theta[axis]>360)
    {
        theta[axis]-=360;
    }
    glutPostRedisplay();
}
void mouse(int btn,int state,int x,int y)
{
```

```
    if(btn==GLUT_LEFT_BUTTON && state==GLUT_DOWN)
    axis=0;
    if(btn==GLUT_RIGHT_BUTTON && state==GLUT_DOWN)
    axis=2;
    if(btn==GLUT_MIDDLE_BUTTON && state==GLUT_DOWN)
    axis=1;
}
void keyboard(unsigned char key, int x, int y){
    if(key == 'x'){
        viewer[0] -= 1;
    }
    if(key == 'X'){
        viewer[0] += 1;
    }
    if(key == 'y'){
        viewer[1] -= 1;
    }
    if(key == 'Y'){
        viewer[1] += 1;
    }
    if(key == 'z'){
        viewer[2] -= 1;
    }
    if(key == 'Z'){
        viewer[2] += 1;
    }
}
void main(int argc,char** argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE| GLUT_DEPTH);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(500,500);
    glutCreateWindow("Spin Cube");
    init();
    glutDisplayFunc(display);
    glutIdleFunc(spin_cube);
    glutMouseFunc(mouse);
    glutKeyboardFunc(keyboard);
    glEnable(GL_DEPTH_TEST);
    glutMainLoop();
}
```

Program 2: Triangle

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


float triangle[3][3]={{350,450,550},{400,573,400},{1,1,1}};
float rot_mat[3][3]={{0},{0},{0}};
float result[3][3]={{0},{0},{0}};
float h=0,k=0,m=0,n=0,theta;
void multiply()
{
    int i,j,l;
    for(i=0;i<3;i++){
        for(j=0;j<3;j++)
        {
            result[i][j]=0;
            for(l=0;l<3;l++)
            {
                result[i][j]+=rot_mat[i][l]*triangle[l][j];
            }
        }
```

```
    }
    for(i=0;i<3;i++){
        for(j=0;j<3;j++)
        {
            printf("%f\t",result[i][j]);
        }
        printf("\n");
    }
}

void rrotate(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 drawtriangle()
{
    glBegin(GL_LINE_LOOP);
    glVertex2f(triangle[0][0],triangle[1][0]);
    printf("Normal Triangle %f %f\n", triangle[0][0],triangle[1][0]);
    glVertex2f(triangle[0][1],triangle[1][1]);
    printf("Normal Triangle %f %f\n", triangle[0][1],triangle[1][1]);
    glVertex2f(triangle[0][2],triangle[1][2]);
    printf("Normal Triangle %f %f\n", triangle[0][2],triangle[1][2]);
    glEnd();
}

void drawrotatetriangle()
{
    glBegin(GL_LINE_LOOP);
    glVertex2f(result[0][0],result[1][0]);
    printf("Rotate Triangle %f %f\n", result[0][0],result[1][0]);
    glVertex2f(result[0][1],result[1][1]);
    printf("Rotate Triangle %f %f\n", result[0][1],result[1][1]);
    glVertex2f(result[0][2],result[1][2]);
    printf("Rotate Triangle %f %f\n", result[0][2],result[1][2]);
    glEnd();
}

void display()
{
    glClearColor(1,1,1,1);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1,0,0);
    drawtriangle();
    rrotate(0,0);
    glColor3f(0,0,0);
    drawrotatetriangle();
    m=h*(cos(theta)-1)+k*(sin(theta));
    n=-k*(cos(theta)-1)-h*(sin(theta));
    printf("m: %f n: %f\n",m, n );
    rrotate(m,n);
    glColor3f(1,0,1);
    drawrotatetriangle();
    glFlush();
}
```

```
void Init()
{
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0,1000,0,1000);
    glMatrixMode(GL_MODELVIEW);

}

void main(int argc, char **argv)
{
    printf("Enter the values for theta, h and k:");
    scanf("%f%f%f",&theta,&h,&k);
    theta=theta*(3.14/180);
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(1000,1000);
    glutCreateWindow("Triangle Rotation");
    Init();
    glutDisplayFunc(display);
    glutMainLoop();

}
```

Program 1: Brenham's

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

int x1,x2,y1,y2;
void myInit()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(0,0,0,1);
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(0,500,0,500);
}

void draw_pixel(int x,int y)
{
    glColor3f(1,0,0);
    glBegin(GL_POINTS);
    glVertex2i(x,y);
    glEnd();
}

void draw_line()
{
    int dx,dy,i,e;
    int incx,incy,inc1,inc2;
    int x,y;

    x = x1;
    y = y1;

    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;

    if(dx>dy)
    {
        draw_pixel(x,y);

        e=2*dy-dx;
        inc1=2*(dy-dx);
        inc2=2*dy;

        for(int i=0;i<dx;i++)
        {
            if(e>=0)
            {
                y+=incy;
                e+=inc1;
            }

            else{
                e+=inc2;
            }
            x+=incx;
            draw_pixel(x,y);
        }
    }

    else
    {
        draw_pixel(x,y);

        e=2*dx-dy;
        inc1=2*(dx-dy);
        inc2=2*dx;

        for(int i=0;i<dy;i++)
        {
            if(e>=0)
            {
                x+=incx;
                e+=inc1;
            }
            else{
                e+=inc2;
            }
            y+=incy;
            draw_pixel(x,y);
        }
    }
}



void myDisplay()
{
    draw_line();
    glFlush();
}

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

    printf("Enter 4 points");
    scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
    printf("(%d,%d) (%d,%d)",x1,y1,x2,y2);
```

```
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB | GLUT_SINGLE);
    glutInitWindowPosition(50,50);
    glutInitWindowSize(500,500);
    glutCreateWindow("Brenham");
    myInit();
    glutDisplayFunc(myDisplay);
    glutMainLoop();
    return 0;
}
```