EXPERIMENT NO: 12

AIM: To implement Sierpinski Gasket using open GL

THEORY:

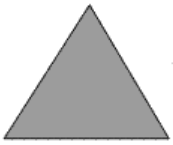
The Sierpinski Gasket is an example of a simple fractal.
One of the ways of constructing it is to start with a triangle and then proceed as follows:

1. Split each edge of the triangle in half to obtain a new point halfway between the edge endpoints. Connect that newly obtained points with each other. This way we have split our original triangle into four smaller ones (one is upside down).

2. Remove the small triangle in the middle.

3. Apply the algorithm to each of the three smaller triangles left.

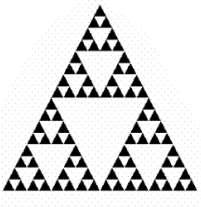
RECURSIVELY DRAWING FRACTAL



Start with a triangle

Connect bisectors of sides and remove central triangle

Repeat



Example with five subdivisions

DRAW ONE TRIANGLE

```
void triangle( GLfloat *a, GLfloat *b, GLfloat *c)

    {glVertex2fv(a);
glVertex2fv(b);
      glVertex2fv(c);     }
```

TRIANGLE SUBDIVISION

```
voiddivide_triangle(GLfloat *a, GLfloat *b, GLfloat *c, int m)
{
    /* triangle subdivision using vertex numbers */
GLfloatv0[2], v1[2], v2[2];
int j;
if(m>0)
    {for(j=0; j<2; j++) v0[j]=(a[j]+b[j])/2;
for(j=0; j<2; j++) v1[j]=(a[j]+c[j])/2;
for(j=0; j<2; j++) v2[j]=(b[j]+c[j])/2;
divide_triangle(a, v0, v1, m-1);
divide_triangle(c, v1, v2, m-1);
divide_triangle(b, v2, v0, m-1);     }
else
triangle(a,b,c);     /* draw triangle at end of recursion */}
```

DISPLAY AND INIT FUNCTIONS

```
void display()
{glClear(GL_COLOR_BUFFER_BIT);
glBegin(GL_TRIANGLES);
divide_triangle(v[0], v[1], v[2], n);
glEnd();
glFlush();}
voidmyinit()
{glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-2.0, 2.0, -2.0, 2.0);
glMatrixMode(GL_MODELVIEW);
glClearColor (1.0, 1.0, 1.0,1.0);
glColor3f(0.0,0.0,0.0);}}
```

CONCLUSION:-Successfully studied and implemented Sierpinski Gasket using open GL

SIGN **GRADE**

DATE

```
/*Sierpienski Gasket
 * This is another simple, introductory OpenGL program.
 * It draws the Sierpinski Gasket on the  window.*/
#include <GL/glut.h>
#include <math.h>

void display(void)
{

 typedef GLfloat point2[2];

 point2 vertices[3] = {{25.0,25.0},{50.0, 75.0},{75.0,25}};


 int i,j, k;

  point2  p = {55, 55};

  glClear (GL_COLOR_BUFFER_BIT);

  /* Compute and display 1000 new points */
   for(k = 0; k <= 4000;k++)
   {
    j = rand()%3; /* pick vertex at random */

 /* Compute halfway point between vertex and p */

     p[0] = (p[0] + vertices[j][0])/2.0;
     p[1] = (p[1] + vertices[j][1])/2.0;

    /* plot the point */

   glBegin(GL_POINTS);
     glVertex2fv(p);
   glEnd();

  }

/* don't wait!
 * start processing buffered OpenGL routines */
   glFlush ();
}
```

```
void init (void)
{
/* select clearing color        */
   glClearColor (0.0, 0.0, 1.0, 0.0);

/* initialize viewing values  */
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 100.0, 0.0, 100.0, -1.0, 1.0);
   glMatrixMode(GL_MODELVIEW);

}

/*Declare initial window size, position, and display mode
 * (single buffer and RGBA).  Open window with "Sierpinski Gasket"
 * in its title bar.  Call initialization routines.
 * Register callback function to display graphics.
 * Enter main loop and process events.*/
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (250, 250); glutInitWindowPosition (100,
   100); glutCreateWindow ("Sierpinski Gasket"); init ();

   glutDisplayFunc(display);
   glutMainLoop();
   return 0;   }
```

Output:-