**Experiment No. 08**

**Fractal Koch Curve**

**Aim:** To implement Fractal (Koch Curve).

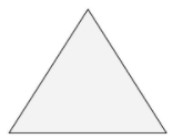
**Theory:** A Koch curve is a fractal curve that can be constructed by taking a straight line segment and replacing it with a pattern of multiple line segments. Then the line segments in that pattern are replaced by the same pattern. Clearly, the generation of a Koch curve is a prime candidate for recursion.

Also, in looking at the the operational definition of a Koch curve, we can see that if we examine the curve between any two vertices (corners) of the curve (for a given, finite iteration level), there are two possible states:

1. The curve is a single straight line between the two points.
2. The curve is a collection of multiple straight lines between the two points.

**Procedure:**

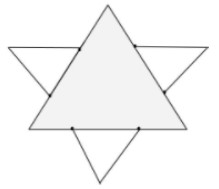
1) Draw an equilateral triangle.



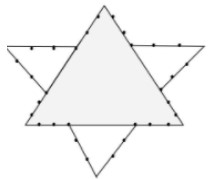
2) Divide each side in three equal parts.



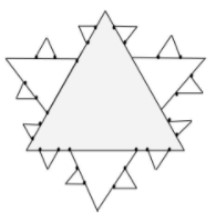
3) Draw an equilateral triangle on each middle part. Measure the length of the middle third to know the length of the sides of these new triangles.



4) Divide each outer side into thirds. You can see the 2nd generation of triangles covers a bit of the first. These three line-segments shouldn’t be parted in three.



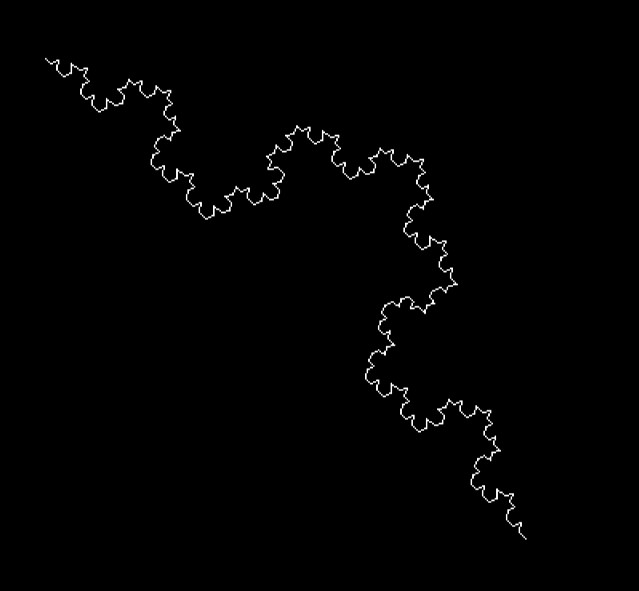
5) Draw an equilateral triangle on each middle part.



**Program:**

| #include<graphics.h>  #include<conio.h>  #include<math.h>  void koch(int x1, int y1, int x2, int y2, int it)  {  float angle = 60\*M\_PI/180;  int x3 = (2\*x1+x2)/3;  int y3 = (2\*y1+y2)/3;  int x4 = (x1+2\*x2)/3;  int y4 = (y1+2\*y2)/3;  int x = x3 + (x4-x3)\*cos(angle)+(y4-y3)\*sin(angle);  int y = y3 - (x4-x3)\*sin(angle)+(y4-y3)\*cos(angle);  if(it > 0)  {  koch(x1, y1, x3, y3, it-1);  koch(x3, y3, x, y, it-1);  koch(x, y, x4, y4, it-1);  koch(x4, y4, x2, y2, it-1);  }  else  {  line(x1, y1, x3, y3);  line(x3, y3, x, y);  line(x, y, x4, y4);  line(x4, y4, x2, y2);  }  }  int main()  {  int gd = DETECT, gm,i,n, x1 = 100, y1 = 100, x2 = 400, y2 = 400;  initgraph(&gd,&gm,"..//bgi");  printf("Enter number of interations");  scanf("%d",&n);  for(i=0;i<n;i++)  {  cleardevice();  koch(x1, y1, x2, y2, i);  getch();  }  return 0;  } |  |
| --- | --- |

**Outcome:**

****

**Conclusion:**

1. Difference from Bezier Curve - The main difference between Bezier and cubic spline is that with a Bezier curve the two of the control points form the end points of the curve and the remaining control points are off the curve. With cubic spline, all the control points are on the curve.

2. Application - The Koch snowflake has been constructed as an example of a continuous curve where drawing a tangent line to any point is impossible.