Generate Koch-snowflake

ADVANCED THEORTICAL METHODS

22PHPH03 BHAGYA R
22PHPH15 SURENDRA KUMAR GOUR
22PHPH18 BIKRAM KESHARI BEHERA

FRACTAL:-

Introduction:-

Benoît Mandelbrot coined the term “fractal” in 1975. The term “fractal” is derived from the Latin word “fractus”, which means "broken" or "fractured". A fractal is defined as a rough or fragmented geometric shape that can be divided into smaller parts and each of the parts is a reduced-size copy (or at least approximately) of the whole (Mandelbrot, 1982; Crownover, 1995).

A mathematical fractal is based on an equation that undergoes iteration, a form of feedback based on recursion (Briggs,1992). Since, fractals are scale invariant, i.e., fractals appear similar at all levels of magnification, fractals are often considered to be infinitely complex objects (in informal terms). Nature is full of fractals. So the concept of fractals has been around for centuries.

Examples of natural objects that are approximate fractals are: various vegetables (cauliflower and broccoli), coastlines, animal coloration patterns, clouds, snowflakes, mountain ranges, lightning bolts. We will also find them throughout the natural world in the patterns of streams, rivers, coastlines, mountains, waves, waterfalls, and water droplets. In the 1960s and 1970s, Mandelbrot began studying complex systems and patterns that exhibited self-similarity at different scales. He coined the term "fractal" to describe these patterns, which he believed could help explain the complex and seemingly random behaviour of natural systems such as weather patterns and stock market fluctuations.

However, not all self-similar objects are fractals. For example, the real line (a straight line) is formally self-similar but fails to have other fractal characteristics. It is irregular enough to be described in Euclidean terms (Crilly, 1991; Edgar, 1990).

A fractal often has the following features:

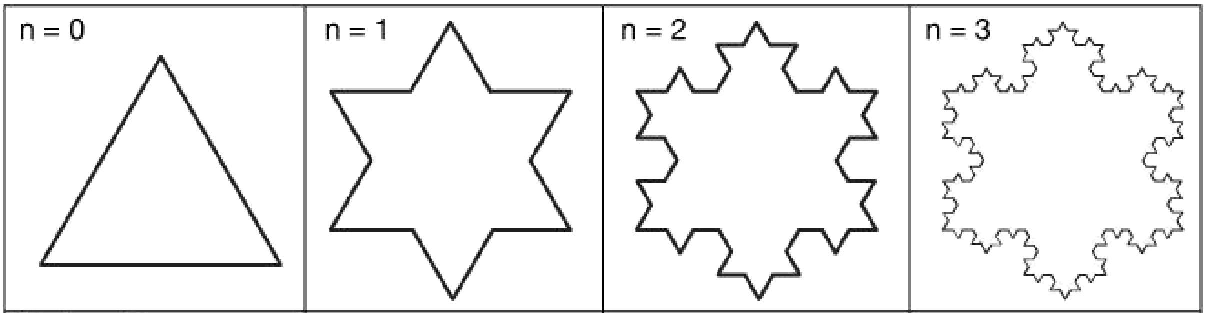
i. A fractal cannot be described by traditional Euclidean geometric language as it is too irregular.
ii. It is self-similar (at least approximately).

iii. It is scale invariant, i.e., It has a fine structure at arbitrarily small scales.
iv. It has dimension in fractions.
v. It has a simple and recursive definition.

Now we are taking One Example of Koch snowflake…

<u>Koch snowflake</u>: -

Mathematically, snowflakes can be described using fractal geometry and the principles of chaos theory. Scientists have used computer simulations to model the growth of snowflakes and study their complex and ever-changing structures.



Number of sides (N)	3	12	48	192
Side length (S)	1	1/3	1/9	1/27
Perimeter length (P)	3	4	5.33	9.11

The Koch snowflake (also known as the Koch curve, Koch star, or Koch island) is a fractal curve. In 1904 Helge von Koch identified a fractal that appeared to model the snowflake. The fractal is built by starting with an equilateral triangle and removing the inner third of each side, building another equilateral triangle at the location where the side was removed, and then repeating the process indefinitely. The process is illustrated above, showing the original triangle at stage 0 and the resulting figures after one, two and three iterations. While the perimeters of the successive stages increase without bound. Consequently, the snowflake encloses a finite area, but has an infinite perimeter.

Each arm of a snowflake is built up from a hexagonal crystal structure, with additional branches and patterns forming as the crystal grows and encounters different temperature and humidity conditions in the atmosphere. The branching patterns of a snowflake are similar, but not identical, on each arm, giving the overall structure a fractal-like quality.

The fractal nature of snowflakes has inspired artists and designers to create fractal patterns and designs in a wide range of media, from snowflake-themed clothing and jewellery to fractal art and architecture. The beauty and complexity of snowflakes continue to capture our imagination and inspire us to explore the wonders of the natural world.

Motivation Behind Study of Koch snowflakes:-

The von Koch curve which has been rigorously analysed and widely used as fractal antenna. The performance of a fractal antenna is directly related to its dimension. So there is a need to propose Koch shapes of lesser dimension and compact size for better performance with compact in size. In 1990's, Nathan Cohen used fractal antenna to rethink about wireless communication. There, aim was to make a compact fractal antenna, which are better than the regular ones. Regular antennas are cut for one type of signal and works for their lengths and certain multiples of their signals wavelengths. But in fractal antennas, as fractal repeat more and more, the fractal antenna can pick up more and more signal, not just one signal. Also as the parameter increases faster than area, fractal antennas can take only quarter of usual space. This enhances the study of fractals.

CONSTRUCTION:-

Let N_n = the number of sides, L_n= the length of a single side, P_n= the length of the perimeter, and A_n= the area of the snowflake

The Koch snowflake can be constructed by starting with an equilateral triangle.

I. Divide the line segment into three segments of equal length.
II. Draw an equilateral triangle that has the middle segment from step 1 as its base and points outward.
III. Remove the line segment that is the base of the triangle from step 2.

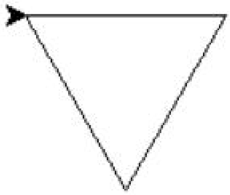
The first iteration of this process produces the outline of a hexagram.

Program Code:-

 # 1st iteration of koch snow-flake:-

```
import turtle
def koch_snowflake(t, order, size):
    if order == 0:
        t.forward(size)
    else:
        for angle in [60, -120, 60, 0]:
            koch_snowflake(t, order-1, size/3)
            t.left(angle)
def main():
    t = turtle.Turtle()
    t.speed(0)
    t.penup()
    t.goto(-100, 50)
    t.pendown()
    for i in range(3):
        koch_snowflake(t, 0, 100)
        t.right(120)
    turtle.done()
if __name__ == '__main__':
```

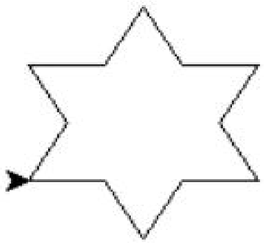
```
main()
```



 # 2nd order iteration koch snow-flake:-

```
#imports turtle library
import turtle
#define the Koch snowflake function with parameters
def koch_snowflake(t, order, size):
    if order == 0:
        t.forward(size)
    else:
        for angle in [60, -120, 60, 0]:
            koch_snowflake(t, order-1, size/3)
            t.left(angle)
#defines the main function with turtle library
def main():
    t = turtle.Turtle()
#sets the turtle speed to 0(fastest speed)
    t.speed(0)
#penup & pendown moves the turtle object which draws the plot
    t.penup()
#sets the co-ordinates for the turtle
    t.goto(-100, 50)
```

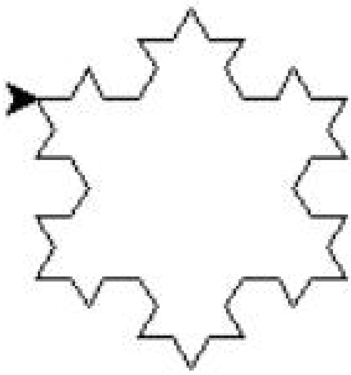
```
    t.pendown()
#loop repeats 3 times with the main function
    for i in range(3):
        koch_snowflake(t, 1, 100)
        t.right(120)
#completes the diagram
    turtle.done()
#ensures that main function runs when the script runs directly
if __name__ == '__main__':
    main()
```



<u># 3rd order iteration</u>

```
import turtle
def koch_snowflake(t, order, size):
    if order == 0:
        t.forward(size)
    else:
        for angle in [60, -120, 60, 0]:
            koch_snowflake(t, order-1, size/3)
            t.left(angle)
def main():
    t = turtle.Turtle()
```

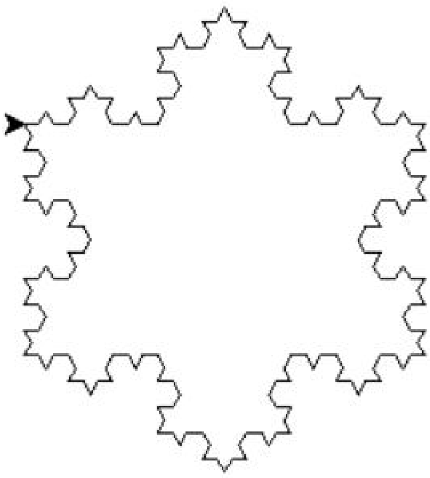
```
    t.speed(0)
    t.penup()
    t.goto(-100, 100)
    t.pendown()
    for i in range(3):
        koch_snowflake(t, 2, 100)
        t.right(120)
    turtle.done()
if __name__ == '__main__':
    main()
```



 # 4th order iteration

```
import turtle
def koch_snowflake(t, order, size):
    if order == 0:
        t.forward(size)
    else:
        for angle in [60, -120, 60, 0]:
```

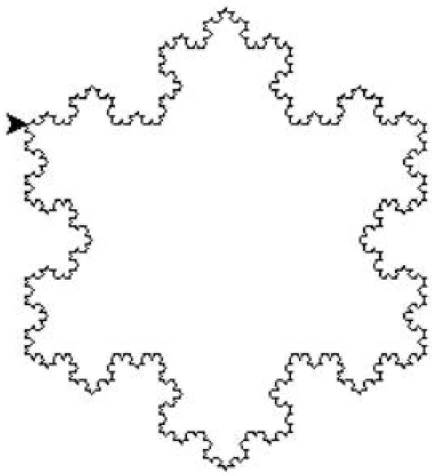
```
            koch_snowflake(t, order-1, size/3)
            t.left(angle)
def main():
    t = turtle.Turtle()
    t.speed(0)
    t.penup()
    t.goto(-50, 100)
    t.pendown()
    for i in range(3):
        koch_snowflake(t, 3, 200)
        t.right(120)
    turtle.done()
if __name__ == '__main__':
    main()
```



 # 5th order iteration

```
import turtle
def koch_snowflake(t, order, size):
    if order == 0:
```

```python
        t.forward(size)
    else:
        for angle in [60, -120, 60, 0]:
            koch_snowflake(t, order-1, size/3)
            t.left(angle)
def main():
    t = turtle.Turtle()
    t.speed(0)
    t.penup()
    t.goto(-100, 50)
    t.pendown()
    for i in range(3):
        koch_snowflake(t, 4, 200)
        t.right(120)
    turtle.done()
if __name__ == '__main__':
    main()
```



Applications: -

Fractals have had a profound impact on many areas of research and technology, including computer graphics, data visualization, and artificial intelligence. They have also inspired artists, musicians, and writers, who have used fractal patterns and concepts in their work to create visually stunning and intellectually engaging creations. The self-similar and fractal nature of the snowflake makes it an ideal candidate for generating complex patterns in digital images and for creating antennas that have a larger surface area and can transmit and receive signals more efficiently. Additionally, researchers have used the snowflake as a model for studying the properties of certain materials.

The applications of fractals are found in every small or large parts of the universe, i.e., from bacteria cultures to galaxies to our body.

A list of application areas of fractals apart from the above mentioned areas are diffusion, economy, fractal art, fractal music, landscapes, Newton's method, special effects, weather, galaxies, rings of Saturn, bacteria cultures, chemical reactions, human anatomy, molecules, Botany, population growth, clouds, coastlines and borderlines, Chemistry, medical science, film industry, wavelet theory, nanotechnology etc. As scientists continue to discover new properties and applications of fractals, it's likely that the Koch Snowflake will continue to play a role in cutting-edge research and innovation.