Fractal Project Report- Sierpinski Carpet and Triangle

Harsha Sreekumar(22PHPH01)
Soumyodip Bandyopadhyay (22PHPH06)

May 4, 2023

1 Introduction

Fractals are a visual representation of repeating patterns or formulas where a simple pattern is repeated multiple times making the pattern progressively complex. A very important characteristic of fractal geometry is its self-similarity where part of the image is very similar to the whole which repeats itself on a smaller and smaller scale.[5]

Figure 1: Romesco Broccoli, an example of self-similarity in fractals

The term fractal was coined by Benoit Mandelbrot, a 20th century mathematician from a latin word fractus which means fragmented or irregular. Such

irregular shapes are seen all around us, leaves, mountains, lightning, etc.

The very earliest application of fractals came when an English Mathematician, Lewis Fry Richardson was studying the length of the English coastline where he stated that the length of the coastline varies with the length of the tool used to measure. Later this paradox of an infinitely long coastline containing a finite area was equated with that of the Koch Snowflake where middle one third length of a line-segment is turned into a triangle repeatedly.

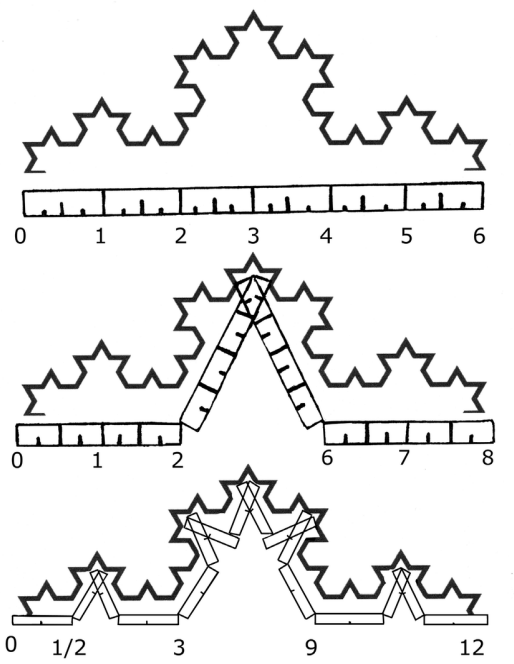


Figure 2: The Coastline Paradox illustrated using Koch Snowflake that shorter measuring units create longer coastlines.

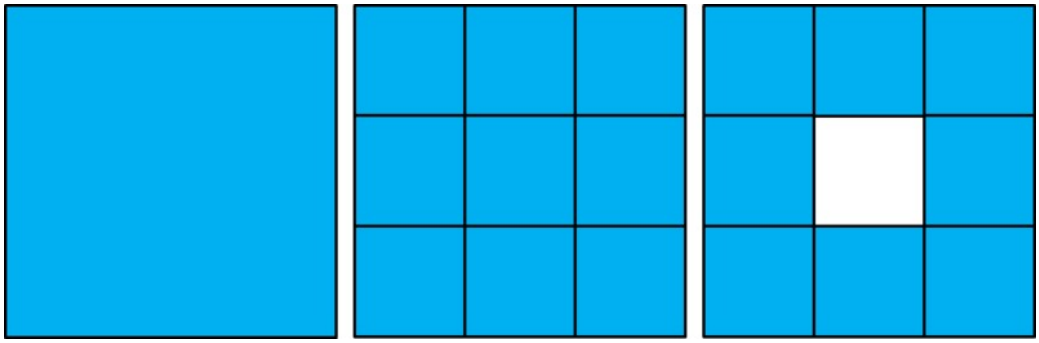
Mandelbrot saw here that the perimeter of such patterns moves to infinity rather than converging to a number and used this example to develop the concept of fractal dimension and hence proving that measuring a coastline is an exercise in approximation.[3]

As mentioned earlier fractals are self-similar and recursive. They are created by iterating as simple mathematical equation or pattern in infinite loops. Also, they have fractal dimension, we are familiar with images of one, two, or three dimension but the peculiarity in fractal geometry is that the patterns have fractional dimensions and it is a measure of the complexity of the pattern.

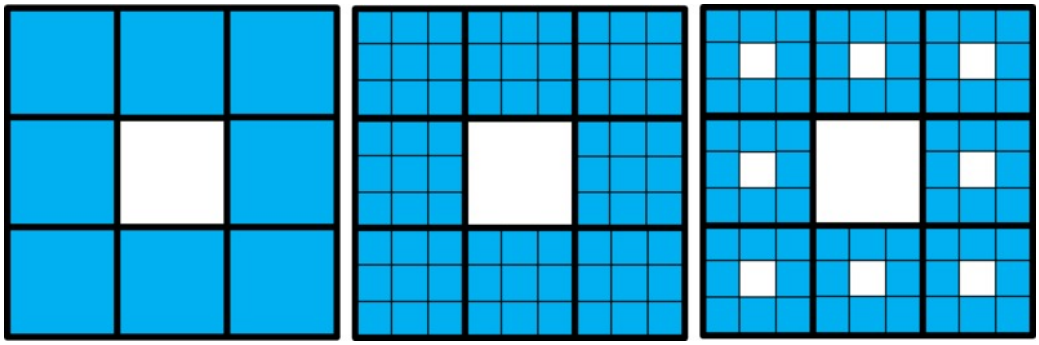
2 Seirpinski Carpet

In 1916, Waclow Seirpinski first described a plane fractal that is the Seirpinski Carpet. It is a generalisation of the cantor set in two dimensions. Following are the steps that is repeated continously inorder to create a Seirpinski Carpet.[2]

- Take a square of area 1 $\implies$ side 1.
  Divide into 9 squares of equal area $\implies$ side of a square= $\frac{1}{3}$.
  Remove the middle square whose area is $\frac{1}{9}$.
  Area of the remaining square= $1 - \frac{1}{9} = \frac{8}{9}$



- Divide each of the square into 9 equal squares $\implies$ side of each square= $\frac{1}{9}$
  Remove the middle square from each of those, area of the newly formed squares= $\frac{1}{81}$
  Area of the figure= $\frac{8}{9} - \frac{8}{81} = \frac{64}{81} = \left(\frac{8}{9}\right)^2$



- Repeating the process n times
  Number of squares= $8^{(n-1)}$
  Area of the figure= $\left(\frac{8}{9}\right)^n$

Area of a Seirpinski Carpet when the process is repeated n times is 0 as $\lim_{n\to\infty} \left(\frac{8}{9}\right)^n = 0$.

The formula for fractal dimension is;

$$n(r) = r^D \tag{1}$$

where r is the scaling factor, $n(r)$ is the number of pieces formed and D is the dimension. In case of Seirpinski Carpet the equation becomes,

$$8 = 3^D \tag{2}$$

$$D = \frac{ln8}{ln3} = 1.8927... \tag{3}$$

2.1 Python Code for Seirpinski Carpet

```
    #Sierpinski Carpet
# importing necessary modules
import numpy as np
from PIL import Image

# total number of times the process will be repeated
total = 7

# size of the image
size = 3**total

# creating an image
square = np.empty([size, size, 3], dtype = np.uint8)
color = np.array([255, 255, 255], dtype = np.uint8)

# filling it black
square.fill(0)

for i in range(0, total + 1):
    stepdown = 3**(total - i)
    for x in range(0, 3**i):

        # checking for the centremost square
        if x % 3 == 1:
            for y in range(0, 3**i):
                if y % 3 == 1:

            # changing its color(iteration changed to fit the whole text)
            square[y * stepdown:(y + 1)*stepdown, x * stepdown:(x + 1)*stepdown] = color

# saving the image produced
save_file = "sierpinski.jpg"
Image.fromarray(square).save(save_file)
```

```
# displaying it in console
i = Image.open("sierpinski.jpg")
i.show()
```

[4] The output of the code is the image shown below;

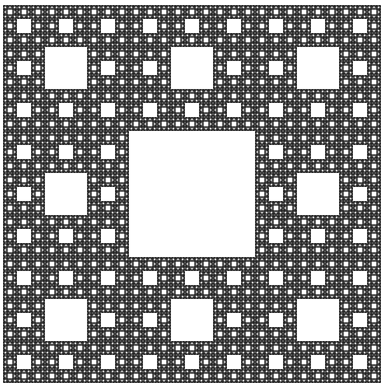


Figure 3: Seirpinski Carpet

3 Sierpinski Triangle

The Sierpinski Triangle is a fractal named after the Polish mathematician Wacław Sierpiński, who described it in 1916. It is also known as the Sierpinski gasket or the Sierpinski sieve.The Sierpinski Triangle is created by starting with a single equilateral triangle and repeatedly dividing it into four smaller equilateral triangles. The central triangle is removed, and the process is repeated with each of the remaining three triangles.Each iteration produces a new triangle with a smaller scale, and the pattern of triangles continues infinitely. The result is a fractal shape that exhibits self-similarity at different scales.

3.1 Area of a Sierpinki triangle

The Sierpinski Triangle is a fractal shape that has an infinite number of iterations, and as such, it does not have a well-defined area.However, we can calculate the area of the Sierpinski Triangle after a certain number of iterations. Each iteration produces three smaller triangles that are each one-fourth the size of the original triangle. Thus, after n iterations, the area of the Sierpinski Triangle is:

$$A_n = \left(\frac{3}{4}\right)^n A_0$$

,

where A_0 is the area of the original triangle.

For example, after 5 iterations, the Sierpinski Triangle has $3^4 = 81$ smaller triangles, and the area of each of these triangles is $(\frac{1}{4})^5 = \frac{1}{1024}$ of the area of the original triangle. Thus, the area of the Sierpinski Triangle after 5 iterations is:

$$A_5 = \left(\frac{3}{4}\right)^5 A_0 = \frac{243}{1024} A_0$$

Note that as the number of iterations approaches infinity, the area of the Sierpinski Triangle approaches zero. This is because the Sierpinski Triangle is a fractal shape that becomes increasingly complex and "spiky" as the number of iterations increases, but the total area remains bounded.[6]

3.2 Fractal Dimension

The Sierpinski Triangle is a classic example of a fractal, which means that it exhibits self-similarity at different scales. The fractal dimension of the Sierpinski Triangle is not an integer, as it is a non-integer value between 1 and 2. The fractal dimension of the Sierpinski Triangle can be calculated using the formula:

$$D = \frac{\log(N)}{\log(S)}$$

,

where N is the number of self-similar copies, and S is the scale factor between the copies. For the Sierpinski Triangle, $N = 3, S = 2$, since each iteration produces three smaller triangles that are each $\frac{1}{4}$ the size of the original triangle.Using this formula, we get:

$$D = \frac{\log(3)}{\log(2)} = \frac{\log(3)}{\log(2)} \approx 1.585$$

This value indicates that the Sierpinski Triangle is a fractal with a dimension that is greater than the dimension of a line (which is 1) but less than the dimension of a plane (which is 2). This non-integer dimension is a hallmark of fractal shapes and allows us to quantify the self-similarity and complexity of the Sierpinski Triangle.

3.3 Python Code for Generating Sierpinski Triangle

```
import turtle

def draw_sierpinski(length, depth):
    if depth == 0:
        # draw a triangle
        for i in range(3):
```

```
            turtle.forward(length)
            turtle.left(120)
    else:
        # recursively draw 3 smaller triangles
        draw_sierpinski(length / 2, depth - 1)
        turtle.forward(length / 2)
        draw_sierpinski(length / 2, depth - 1)
        turtle.backward(length / 2)
        turtle.left(60)
        turtle.forward(length / 2)
        turtle.right(60)
        draw_sierpinski(length / 2, depth - 1)
        turtle.left(60)
        turtle.backward(length / 2)
        turtle.right(60)

# set up the turtle
turtle.speed(0)
turtle.penup()
turtle.goto(-200, -200)
turtle.pendown()

# draw the Sierpinski Triangle
draw_sierpinski(400, 5)

# exit on click
turtle.exitonclick()
```

In this code, we use the Turtle graphics library to draw the Sierpinski Triangle.We also use Turtle commands to move the turtle around and draw the three smaller triangles. The functions

```
draw_sierpinski
```

function takes two arguments: the length of the side of the triangle and the depth of recursion. When the depth is zero, we draw a triangle using the Turtle commands.Otherwise, we recursively call

```
draw_sierpinski
```

with a smaller length and a depth that is one less than the current depth. We also use Turtle commands to move the turtle around and draw the three smaller triangles.

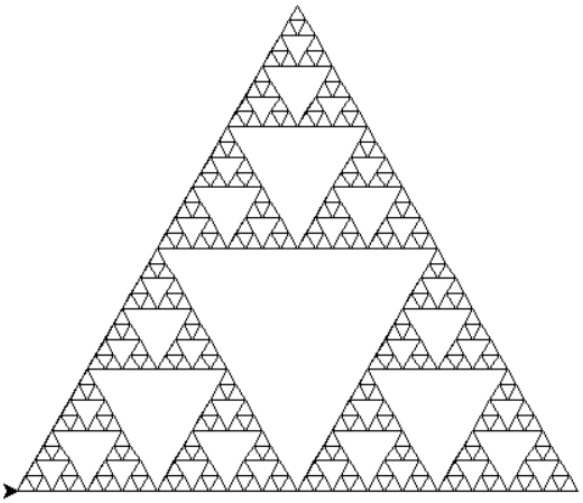


Figure 4: Sierpinski Triangle

This is the output of the above code.

References

[1] Sierpiński carpet — Illustrative Mathematics. https://tasks.illustrativemathematics.org/content-standards/tasks/1523, 2023. [Online; accessed 30-April-2023].

[2] Sierpiński carpet — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Sierpi%C5%84ski_carpet&oldid=1147155003, 2023. [Online; accessed 30-April-2023].

[3] Craig Haggit. How fractals work — How stuff works. https://science.howstuffworks.com/math-concepts/fractals.htm, 2023. [Online; accessed 30-April-2023].

[4] Pulkit Singh. Python, sierpinski carpet. https://www.geeksforgeeks.org/python-sierpinski-carpet/, 2023. [Online; accessed 30-April-2023].

[5] Steven H Strogatz. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering.* CRC press, 2018.

[6] Wikipedia contributors. Sierpiński triangle — Wikipedia, the free encyclopedia, 2023. [Online; accessed 3-May-2023].