AMERICAN Journal of Science on Integration and Human Development
Volume 2, Issue 1, 2024 ISSN (E): 2993-2750

Building Fractal Shapes with Python Programming Language

Khurramova Rano Ibragimovna
Associate Professor of the Department of Convergence of Digital Technologies of the Tashkent University of Information Technologies named after Muhammad al-Khwarizmi, Ph.D.

Egamberdiyeva Nodirabegim Abdunamid qizi
Master of Tashkent University of Information Technologies named after Muhammad al-Khwarizmi

Abstract: This article provides information about the concept of fractals, properties of fractals and software that generates fractals. It also covers algorithm of building one of the famous fractals, the Koch Snowflake, and the process of visualizing it using the Turtle library of the Python programming language.

Keywords: fractal, properties of fractals, fractal dimension, fractal generators, Mandelbulb 3D, Apophis, FractalNow, Koch Snowflake, Python, Turtle Library.

Today, the study of the mathematical aspects of the fractal theory, as well as the methods of describing natural processes and phenomena using the ideas of the fractal theory, is an independent new field of science. It has expanded to such an extent that it is divided into several narrow specializations. The theory of fractals has become a link between sciences. Interest in the study of processes serving the fractal geometry of nature led to the emergence of new scientific directions in physics, mathematics, biology, material science and other sciences. The approach of different scientific directions based on a single structure is not accidental, but the result of the properties of the fractal structure.

Fractals model complex physical processes and dynamic systems. The basic principle of fractals is that a simple process that goes through an infinite number of repetitions becomes a complex process. Fractals attempt to model a complex process by looking for the simple process beneath.

Most fractals work on the principle of a feedback loop. A simple operation is performed on a piece of data and then reinserted. This process is repeated an infinite number of times. The boundary of the produced process is fractal.

Almost all fractals are at least partially self-similar. This means that a part of a fractal is the same as the whole fractal itself, just smaller.

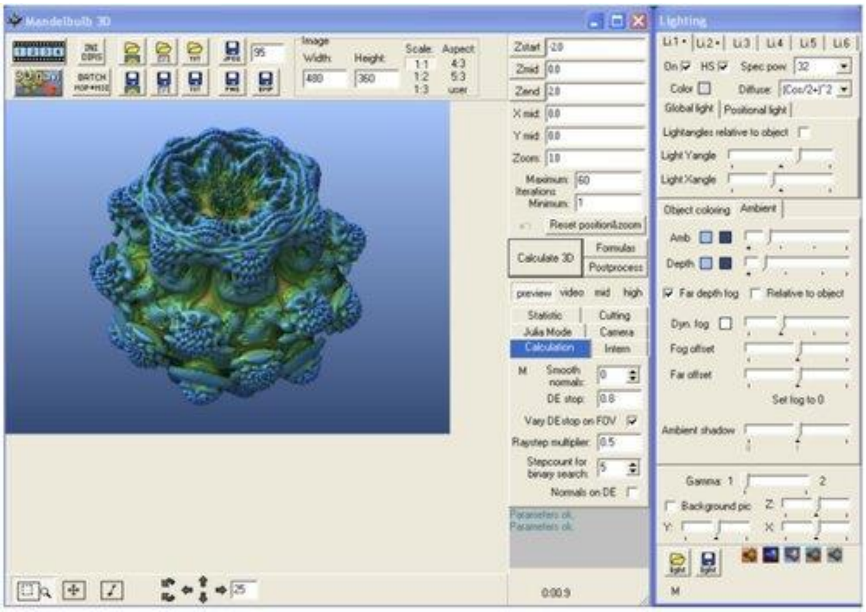
Fractals can look very complicated. However, they are usually very simple processes that lead to complex results. This feature carries over to chaos theory. If an event has complex outcomes, it does not mean that it has a complex starting point.

Fractal dimensions are used to measure the complexity of objects. Fractal theory makes it possible to define measurement methods for issues that are traditionally meaningless or impossible to measure..

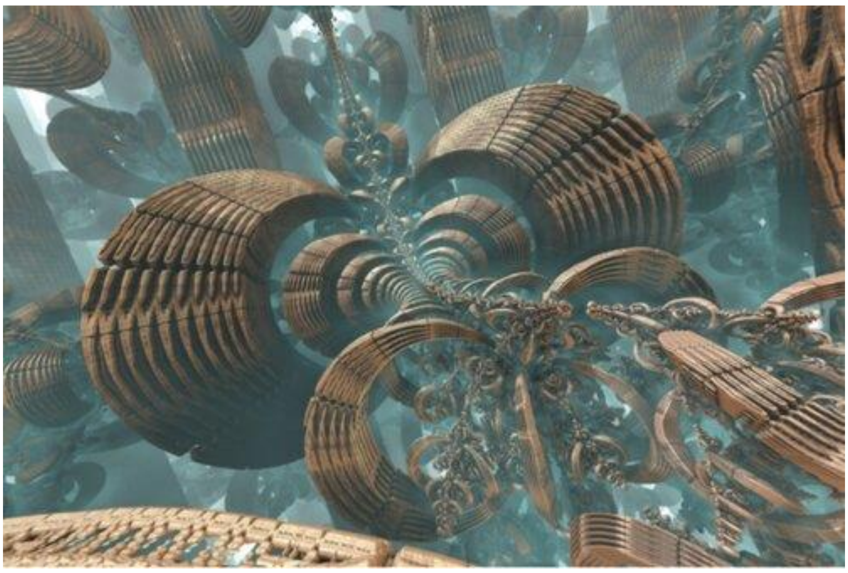
Fractal research is one of the most widely developed fields today. There are several instruments for graphical representation and visualization of fractals through computer technology, and Table 1 lists 3 of the most common ones.

1-table. Instruments that generate fractal shapes

№	Program name	Program description and features
1	Mandelbulb 3D	Free 3D fractal generating program available for the Windows operating system that is convenient for creating various fractal objects from non-linear equations. As a 3D rendering application, it features shadow and glow effects, color, depth of field, and specularity.
2	Apofiz	Open source fractal generator that allows you to edit, create and visualize fractal flames. The different components of the fractal can be accessed directly through the scripting language, and the color and position can be adjusted. Random edits can be applied to images using the program's mutation window.
3	FractalNow	Free and open source fractal generator that can be used to create various types of geometric images. It combines a command-line and graphics tool called QFractalNow to easily implement advanced algorithms and create custom images..



1-figure. Mandelbulb 3D fractal generation window [6].

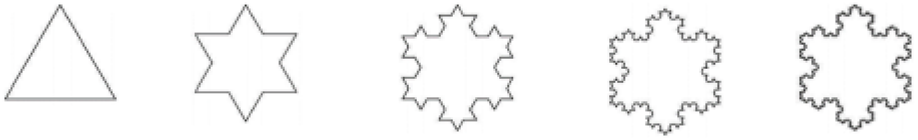


2-figure. Fractal generated by Mandelbulb 3D [6].

In this article, we will look at the process of creating a Koch snowflake and programming it using the Python "Turtle" library

Building a Koch Snowflake fractal

The famous German mathematician von Koch came across the description of curves in 1904 while studying the work of Georg Cantor and Karl Weierstrass. Even an insignificant small segment of the curve exactly reproduces the characteristics of the curve. Koch began to build his "personal" line, forever entered into the annals of mathematics under the name "Koch Snowflake" [1].



3-figure. Koch snowflake and its construction [3]

A snowflake is created by iteratively repeating the following sequence of steps, starting with an equilateral triangle:

1. Dividing the line segment into 3 of equal length
2. Draw an equilateral triangle facing up to the middle segment
3. Remove the midline segment (the base of the triangle)

Programming Kox Snowflake Construction Using Python Programming Language

We use a Python library called "Turtle" to visualize this fractal. First, we should import library:

```
import turtle
```

Let's start creating a Cox snowflake by creating a function for the Cox curve. A recursive function is selected as the function type. The parameters of the function are as follows:

1. turtle - is a point that draws a fractal
2. divis - the number of executions of the above-mentioned actions
3. size - the length of the sides of an equilateral triangle

Inside this function is a key register that advances the turtle by the length of one side when the number of divisions reaches 0. However, if the divisions are greater than 1, we have a for loop that uses the Lindenmayer system to draw the fractal.

The Lindenmayer system is a way to represent a recursive fractal as a string. We use the following string shapes and rules to create a Koch snowflake:

- initial line "FFF"
- line rewriting rule "F" → "F+F-F+F", at an angle of 60 °.
- "F" - forward movement,
- turn right with "+" angle,
- "-" turn left.

Based on these rules, we consider the following recursive function:

```
def kox_fractal(turtle, divis, size):
        if(divis == 0):
```

```
            turtle.forward(size)
    else:
            for angle in [60, -120, 60, 0]:
                    Kox_fract(turtle, divis - 1, size / 3)
                    turtle.left(angle)
```

After creating the function, we set the following variables for the function's access parameters:

```
divis = 10
more detailed the diagram
size = 2000
```

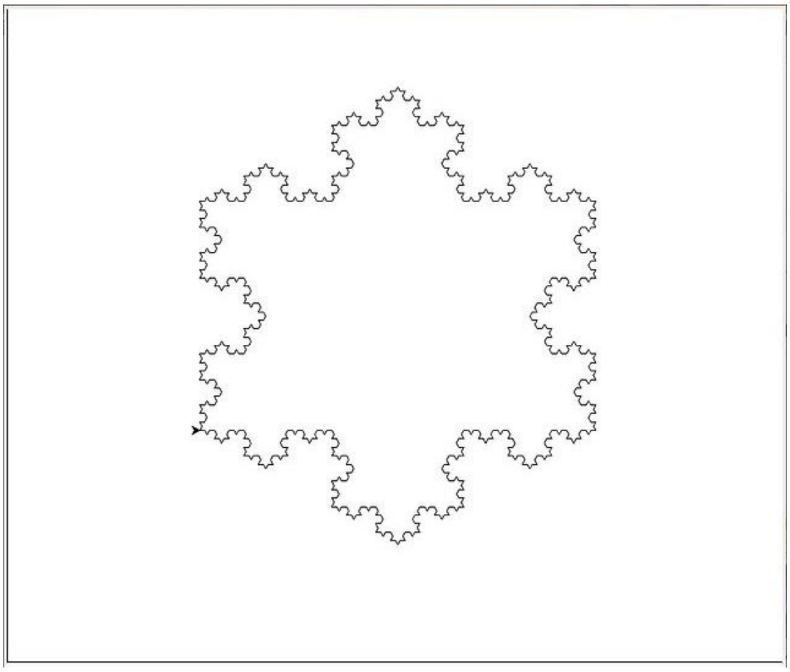
To visually draw a fractal, we refer to the methods of the "Turtle" library:

```
wn = turtle.Screen()
wn.setup(width=1000, height=500)
turtle.speed(100)
turtle.pendown()
```

Call the Koch curve function 3 times and each time rotate the turtle 120° counter-clockwise.

```
for i in range(0, 3):
 kox_fractal(turtle, divis, size)
 turtle.left(-120)
```

After running the program, we can get the following final result.



4-figure. Koch Snowflake (result obtained via the Python "Turtle" library)

In conclusion, we can say that the Koch snowflake fractal was modeled in the article, and the fractal visual was expressed through the Turtle library of the Python programming language through this model. It was considered that any fractal shape can be modeled using the analytical dependence of geometric parameters on iteration, and the surface area of the fractal shape tends to infinity.

References

1. Mandelbrot B.B. Fractal geometry of nature. - M.: Institute of computer research, 2002. C - 656.
2. Перерва Л.М., Юдин В.В., П 27 Фрактальное моделирование.: tutorial / Vladivostok: ВГУЭС, 2007. – 186 с
3. Anarova Sh.A. Fraktallar nazariyasi va fraktal grafika. Book. -T.: "Universitet", 2021. 289-b.
4. Nazirov Sh.A., Anarova Sh.A., Nuraliyev F.M. Fraktallar nazariyasi asoslari. Monograph. – T.: Navruz nashriyoti, 2017. 128-b.
5. https://medium.com/nerd-for-tech/programming-fractals-in-python-d42db4e2ed33
6. https://aiartists.org/fractal-art-generators