Final Project Report

Sierpinski Triangle and Koch Snowflake Fractals by Python and Pgfplots

Bursa Technical University
Department of Mechanical Engineering
Computer Programming Course

Team 0

Name	Student ID
Levent Aydinbakar	1234567890
Levent Aydinbakar	1234567890

Instructor: Dr. Levent Aydinbakar

Abstract

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

December 28, 2022

Contents

List of Figures

List of Tables

1 Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris. [1].

2 Sierpinski Triangle

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum. [2].

2.1 Sierpinski triangle by random points

Sierpinski triangle can be obtained by randomly selected points. To draw it in this way one can follow the path below.

2.1.1 Three corner points of an equilateral triangle

An equilateral triangle is represented by its corner points first.

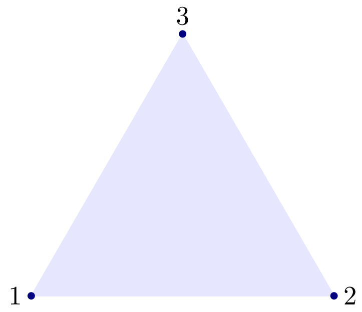

Figure 1: An equilateral triangle corner points

2.1.2 Selecting a random point

Later, a random point is choosen on the triangle.

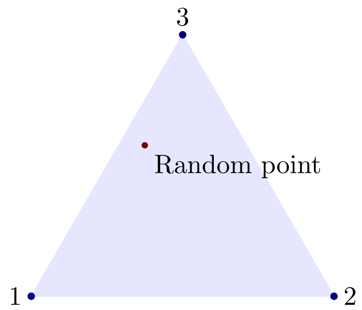

Figure 2: Random point on the equilateral triangle

2.1.3 Selecting a midpoint

In the following step, a corner point is selected randomly and another point is drawn in the middle of this corner point and the random point (see Figure 3).

When this process is repeated for 10 times, 100 times, 100 times, 1000 times, and 20000 times, Figure 4 is obtained.

2.1.4 Method

Figure 4 and other fractal pictures can be obtained with some simple calculations in Python. There are also some libraries to do it. However, an example script can generate the Sierpinski Triangle manually is given in Appendix A.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante.

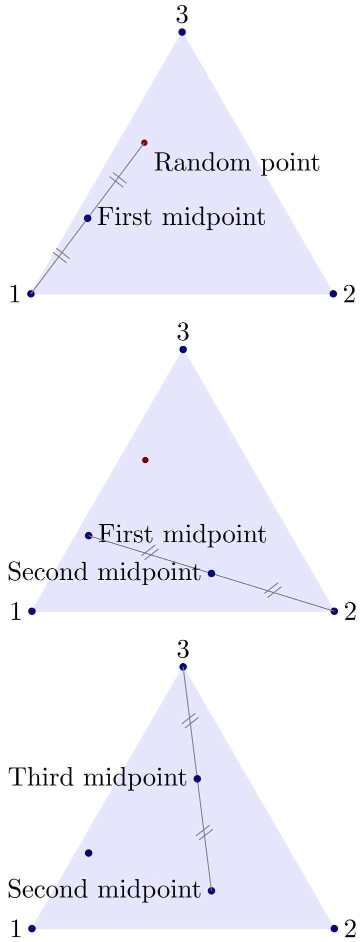

Figure 3: The first midpoint is selected in the middle of the point 1 (randomly selected) and the random point (*top*). The second midpoint is selected in the middle of the point 2 (randomly selected) and the first midpoint (*middle*). The third midpoint is selected in the middle of the point 3 (randomly selected) and the second midpoint (*bottom*)

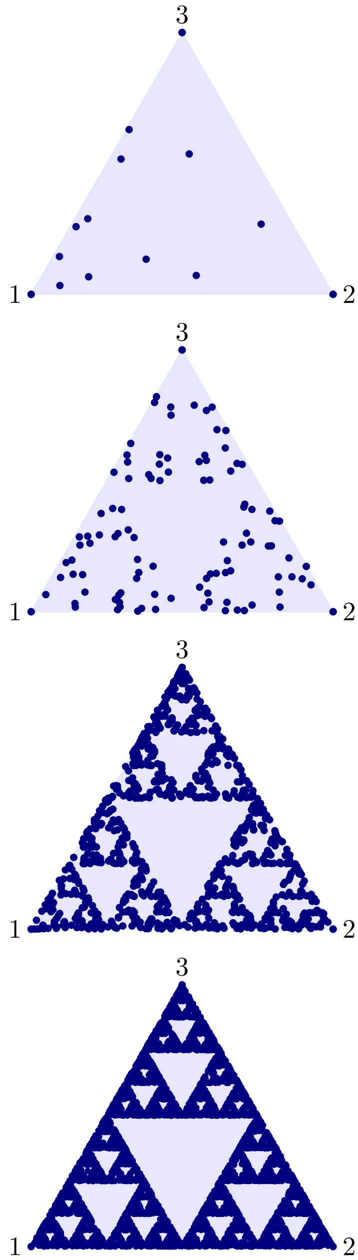

Figure 4: The points after 10, 100, 1000, and 2000 iterations from *top* to *bottom*

2.2 Upside down Sierpinski triangle

Another way of making Sierpinski triangle is explained in the following sections.

2.2.1 Start with a triangle

Let's use a upside down equilateral triangle first. Then, take its half and recombine three of the half models together to obtain the original shape (see Figures 5, 6, and 7).

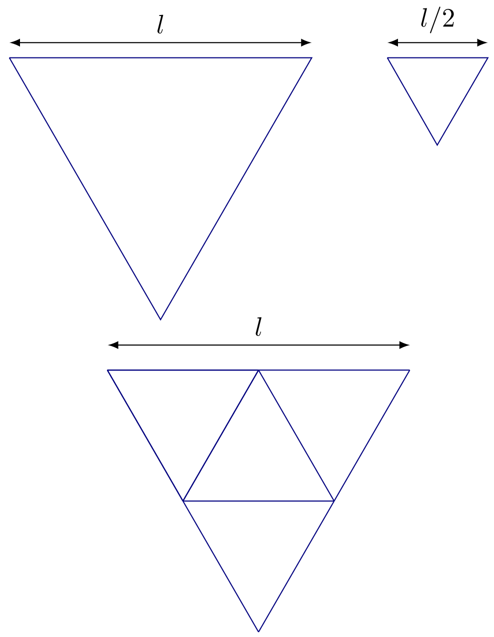

Figure 5: The upside down equilateral triangle, halved model, and recombined model, from *left* to *right* and *top* to *bottom*

The scripts can be seen in Appendix X.

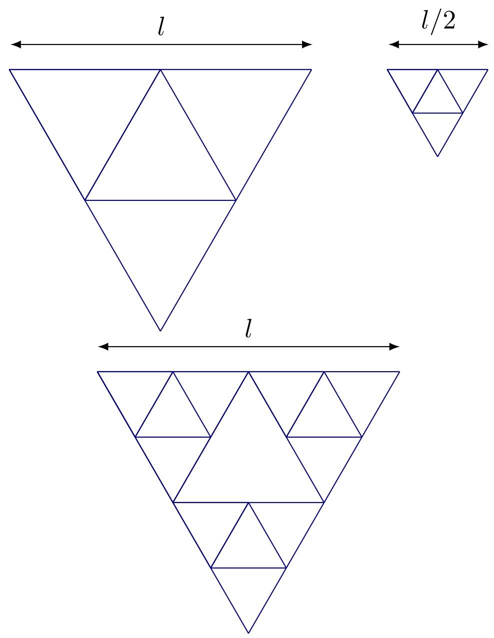

Figure 6: The equilateral triangle at the second step, halved model, and recombined model from *left* to *right* and *top* to *bottom*

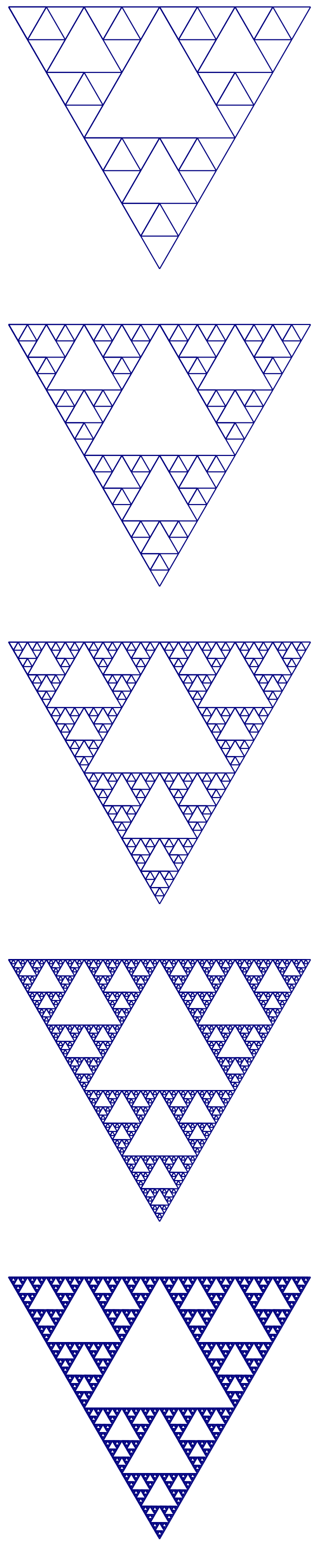

Figure 7: The third, fourth, fifth, sixth, and seventh steps from *top* to *bottom*

3 Koch Snowflake

In Section 2, we show how to make the Sierpinski Triangle. Here we will show the Koch Snowflake [3, 4, 5].

The Koch Snowflake can also be obtained from a equilateral triangle.

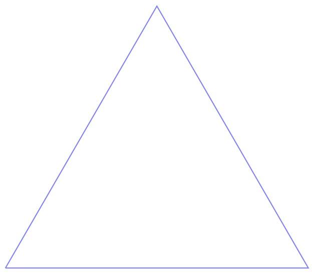

Figure 8: An equilateral triangle corner points

Then the lines are divided into 3 equal parts, the middle part is rotated 60° around both of the points.

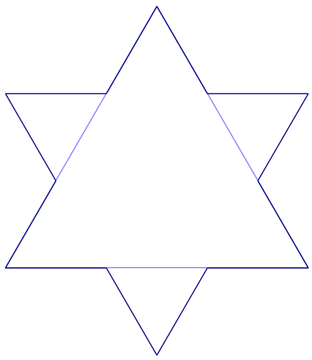

Figure 9: A first order Koch Snowflake

When we continue dividing each line and rotating the midline, we can obtain Figure 10.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

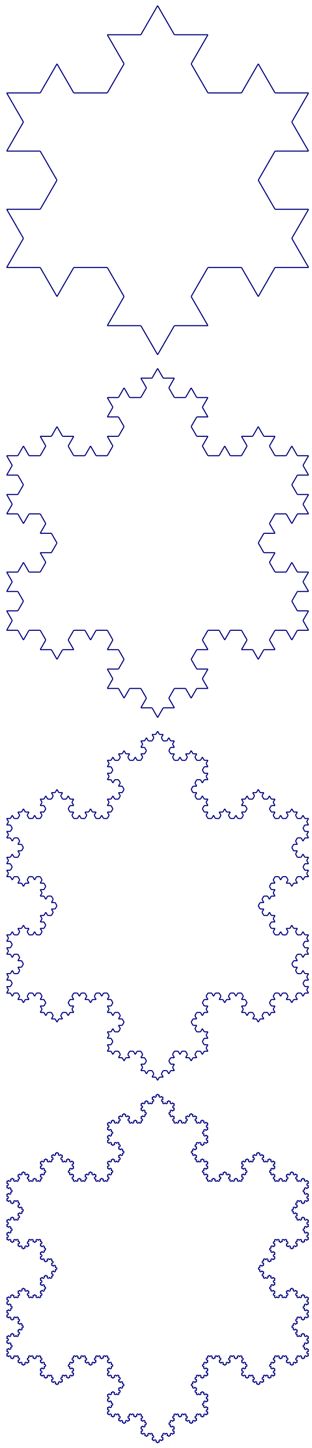

Figure 10: A second, third, fourth, fifth, and sixth order Koch Snowflakes from from *top* to *bottom*

4 Fractals with Pgfplots

There is a library of Pgfplots to draw fractals. For example see Appendix B for the script to draw Figure 11

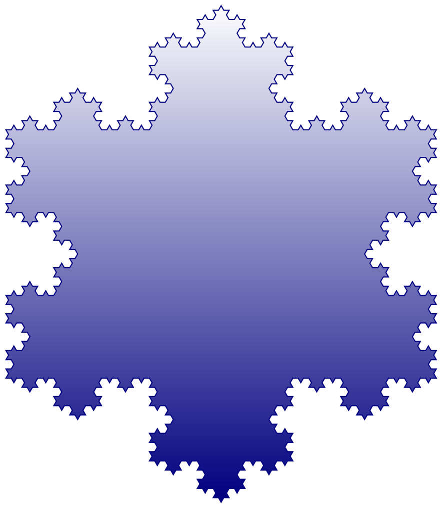

Figure 11: A Koch Snowflake drawn using Pgfplots

Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis.

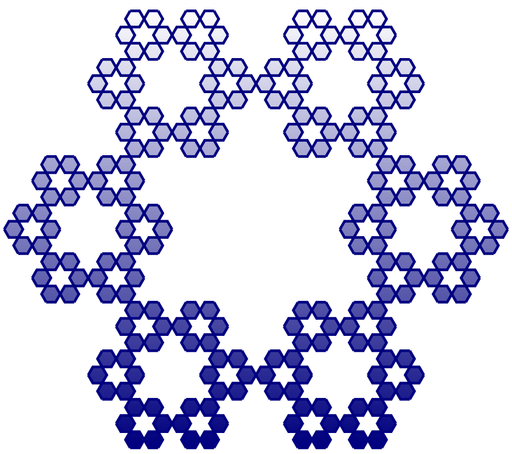

Figure 12: A hexagon fractal drawn using Pgfplots

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo.

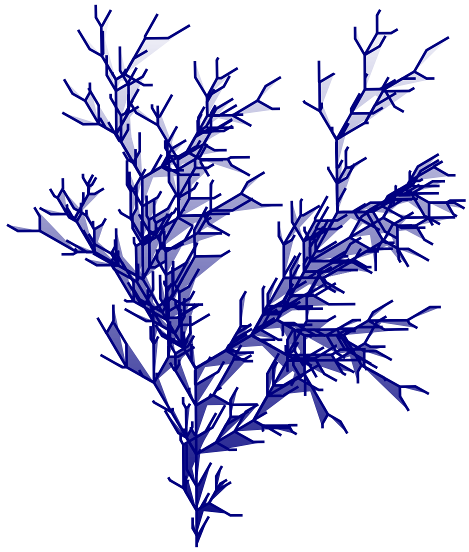

Figure 13: A tree drawn using Pgfplots

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Table 1: Contribution of each team member

Part	Member(s)	Contribution rate
Python scripts	Levent Aydinbakar	70 %
Python scripts	Other Student	30 %
Shell scripts	Levent Aydinbakar	100 %
Gnuplot scripts	Levent Aydinbakar	100 %
Pgfplots graphs	Levent Aydinbakar	100 %
Tikz sketches	Levent Aydinbakar	100 %
LaTeX scripts	Levent Aydinbakar	100 %
Research of the subject	Levent Aydinbakar	100 %

5 Contibutions

Contributions of each team member in this report is given in Table 1.

6 Conclusions

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

References

[1] What are fractals? https://fractalfoundation.org/resources/what-are-fractals/. Accessed on 26 December 2022.

[2] Sierpinski tirangle. https://en.wikipedia.org/wiki/Sierpinski_triangle. Accessed on 26 December 2022.

[3] Michel L Lapidus and Michael MH Pang. Eigenfunctions of the koch snowflake domain. *Communications in mathematical physics*, 172(2):359–376, 1995.

[4] Orhan Pamuk. *Snow*. Vintage, 2006.

[5] Levent Aydinbakar. *A Comparative Turbomachinery Flow Study with the Conservative and Nonconservative Forms of the Space–Time Variational Multiscale Method in the Inertial and Non-inertial Reference Frames*. PhD thesis, Waseda University, 2021.

A Sierpinski Triangle by Python

```
#!/usr/bin/env python3
# Import necessary modules
import numpy
import math
import random
import argparse

# Argument parser to read info from commandline
parser = argparse.ArgumentParser()
parser.add_argument("-i","--iterations", \
                    default=100, type=int, \
                    help="Number of iterations (=100).")
args = parser.parse_args()

# Define length
l=4

# A function to rotate a point around another point
def rotate(point, origin, degrees):
  radians = numpy.deg2rad(degrees)
  x,y = point
  offset_x, offset_y = origin
  adjusted_x = (x - offset_x)
  adjusted_y = (y - offset_y)
  cos_rad = numpy.cos(radians)
  sin_rad = numpy.sin(radians)
  qx = offset_x + cos_rad * adjusted_x - sin_rad * adjusted_y
  qy = offset_y + sin_rad * adjusted_x + cos_rad * adjusted_y
  return qx, qy

# Make the triangle
p1=numpy.array([0,0])
p2=numpy.array([1,0])
p3=rotate(p2, p1, 60)

# Put a random point
random_point=numpy.array([1.5,2])

# Make a tuple to select p1, p2 or p3 randomly
r=numpy.stack((p1,p2,p3))

# Add the first point
p=numpy.vstack((p1,p2,p3,(p1+random_point)/2))

# Add the other points
for i in range(args.iterations):
  pp=random.choice(r)
  pnew=(pp+p[-1])/2
  p=numpy.vstack((p, pnew))

# Save the points to a file
# Add header x,y for Tikz
with open("sierpinski_points.txt","w") as f:
  f.write("x,y\n")
  numpy.savetxt(f, p, fmt="%1.5f", delimiter=",")
```

B Koch Snowflake by Pgfplots

```
\documentclass[tikz,margin=2mm]{standalone}
\usepackage{tikz}

\usetikzlibrary{lindenmayersystems}

\begin{document}%
\def\hexagwidth{4cm}%
\pgfdeclarelindenmayersystem{Sierpinski hexagon}{
  \symbol{X}{\pgflsystemdrawforward}
    \symbol{Y}{\pgflsystemmoveforward\pgflsystemmoveforward\pgflsystemmoveforward}
    \rule{X -> X+X+X+X+X+X+Y}
    \rule{Y -> YYY}
}%
\foreach \level in {4,...,4}{%
\tikzset{
    l-system={step=\hexagwidth/3^\level, order=\level, angle=60}
}%
\begin{tikzpicture}
%  \fill[blue!50!black] (0,0) l-system
\shadedraw [top color=white, bottom color=blue!50!black, draw=blue!50!black]
l-system
  [l-system={Sierpinski hexagon, axiom=X}] ;
\end{tikzpicture}
}%
\end{document}
```