



Complex Systems

2022/2023

Exercise Sheet 8

Ernesto Costa
Tiago Baptista

8

Fractals

8.1 Exercises

Exercise 8.1 E

The Pascal's triangle is constructed as follows. The top and edges of the triangle are filled with 1's. Other entries are obtained by adding the two entries directly above them. Sketch the first 15 rows of the Pascal's triangle. Now shade all the cells that have an odd number in them. Does the resultant image look familiar?

■

Exercise 8.2 M

Write a software program that displays the image from the previous exercise. To do that, compute the triangle row by row, and plot a dot whenever the number of the entry is divisible by 2. Use this program and test other settings (e.g. if the number is divisible by 3).

■

Exercise 8.3 E

Determine the dimension of the Sierpinski carpet, shown in figure 9.1.

■

Exercise 8.4 E

Determine the dimension of the Menger sponge, shown in figure 9.2.

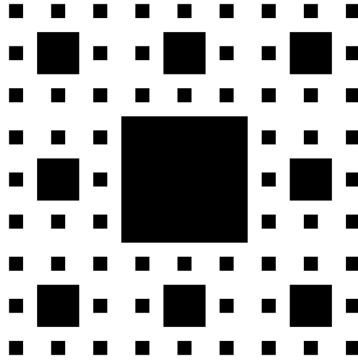


Figure 8.1: The Sierpinski carpet.

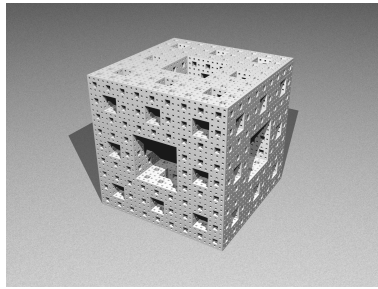


Figure 8.2: The Menger sponge. (Image source: Amir R. Baserinia, <http://en.wikipedia.org/wiki/File:Menger.png>, licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.)

8.2 The Chaos Game

The `simcx` framework has a builtin simulator (`simcx.simulators.IFS`) that allows the creation of fractals using the chaos game. Using the `Points2D` visual, we can visualize the image created. Here is an example of its use, to create the Sierpinski Triangle.

The `IFS` class receives the list of transformations to use, and another list with the probability associated with each of these transformations. We use transformations from the `matplotlib.transforms` module.

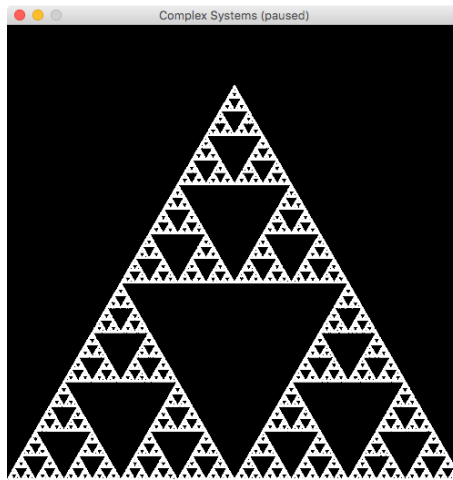


Figure 8.3: The Sierpinski triangle. Generated in simcx using the IFS simulator.

```
t1 = Affine2D().scale(1/2, 1/2)
t2 = Affine2D().scale(1/2, 1/2).translate(1/2, 0)
t3 = Affine2D().scale(1/2, 1/2).translate(1/4,
    3*0.5/4)

transforms = [t1, t2, t3]
probs = [1/3, 1/3, 1/3]
```

Now we can create the simulator, visual, and launch the simcx framework.

```
# Create the simulator
# The step size indicates the number of points
#   computed for each step of the simulator
sim = simcx.simulators.IFS(transforms, probs,
    step_size=100)

# Create the visual, and connect to the simulator
vis = simcx.visuals.Points2D(sim)

# Create the display, attach simulator and visual,
#   and run
display = simcx.Display()
display.add_simulator(sim)
display.add_visual(vis)
simcx.run()
```

This code will, after some iterations, yield the image shown in figure 9.3.

Exercise 8.5 M

Using the IFS simulator in the simcx framework, create the following fractal images:

1. The Sierpinski carpet.
2. The Koch curve.
3. The Koch snowflake.

■

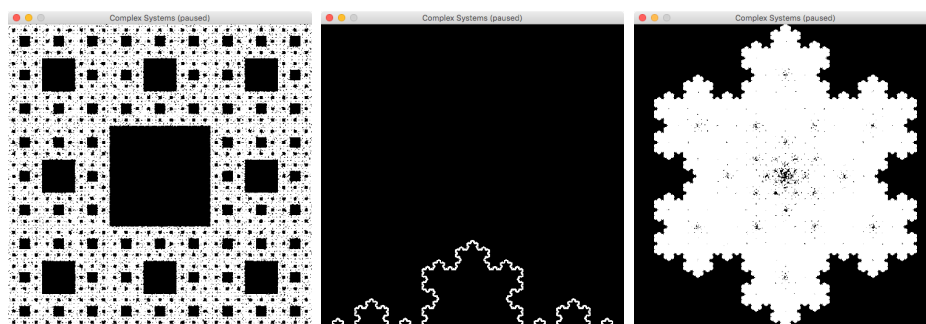


Figure 8.4: Renderings of several fractals using the simcx IFS simulator.

Further Reading

- ✓ David P. Feldman, *Chaos and Fractals – An Elementary Introduction*.