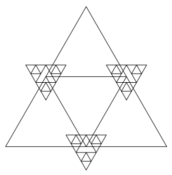
Fractals and the Beauty of Nature
DM550 - Fall Project 2017

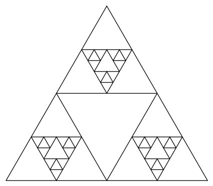
Chanthosh Sivanandam
Erik Andersen
Henrik Flindt

October 25, 2017

1 Sierpinski Triangle

Initially, implementation of the Sierpinski triangle was build upon the idea of placing inverted triangles inside other triangles. It turned out to be a tad complicated, but it yielded some rather interesting results, which can be seen here:


(a) An unsuccessful attempt


(b) Another unsuccessful attempt

After meddling around for some time, a decision was made to try the approach suggested in the project description. A successful algorithm that gave the expected result was then rapidly developed.

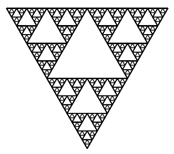


Figure 2: Sierpinski triangle with 5 subdivisions.

Revisiting the initial code, the faultiness of its algorithm became apparent and hence could be corrected. The effectiveness of this new code compared to the approach recommended in the project description is due to the lack of redundant drawing.

We used an iterative development process, meaning we started out by making a small piece of the code work in one iteration. Through testing, trial and error the goal of the iterative step was reached and we then carried on with the next step, where more code was implemented and tested. After several steps, we realized that our approach was overly complicated, so we started again from scratch. This time we did not care for optimization of the algorithm, and we solely focused on correctness.[1]

[1]The source code for this part of the project can be found in the file sierpinsky-triangle.py

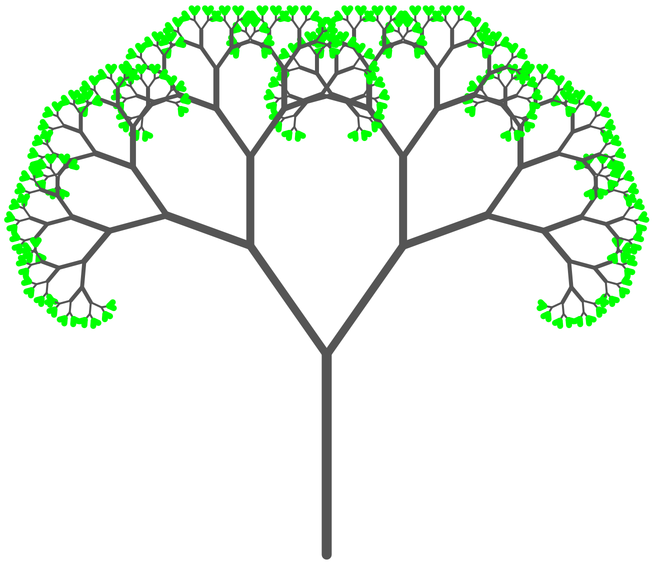


Figure 3: Binary tree

2 Binary three

2.1 Specification and design

We began by trying to define a function with two parameters (size, step)

3 Fern

First of all we define a function for the stem on the fern with several important parameters (length, width, segments curve, steps) The segments are the number of branches on the fern. To demonstrate segments the for loop is necessary tool. The pensize is thickness of the leaves and for each step the pensize degrees with 0,3. The curve is the angle of the leaves and for each turn the angle degrees with 0,3. One of the interesting functions in python is the clone method. The idea with a clone method is too clone every time we turn left or right. The function calls itself with the parameters and the clone but this time the length and width degrees with 0,3 and the segments degrees with one step.

4 FDL-Parser

Since the fdl-file is to be read by an instance of the turtle module's Turtle class, it was decided to make a subclass of it called SmartTurtle. This class should be able to read a fdl-file, and execute he commands, giving a reasonable result. Through analyzis of several fdl-files, a decision as to which attributes the SmartTurtle class should implement could be made. Thus, an instance of the SmartTurtle should have a start attribute containg a command, a length, a depth, a dictionary of rules, and one of commands which could be called with appropriate arguments. The rules should be unfolded according to the depth of the smartturtle, so a unfolded attribute would be convenient. This attribute would initially be an empty string, but when the user of the SmartTurtle class would want to execute the commands in the fdl-file loaded, a function to unfold the rules list and store them as a string in the unfolded attribute, will be required. This functionality is implemented in SmartTurtle's step function.

```
  public class FOO{
  static void int main(String[] args){
    System.out.println(``Hello Latax'');
  }
}
```