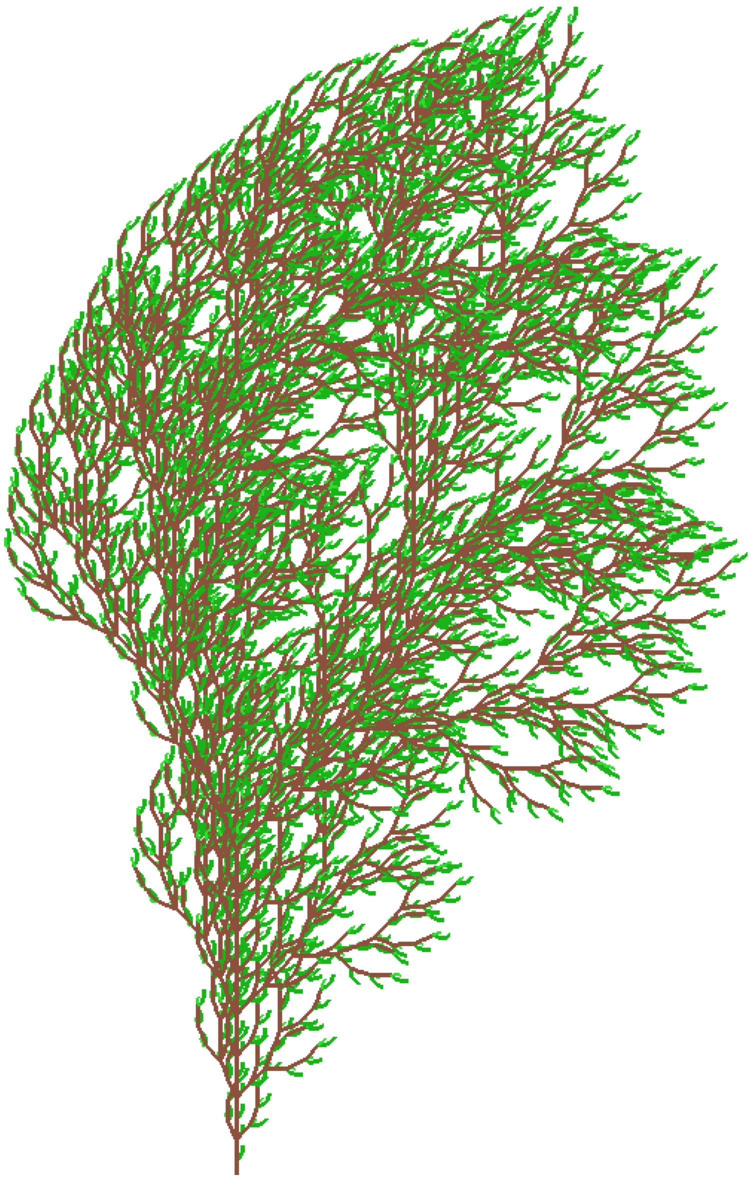
LINDENMAYER-SYSTEMS

2D Implementation of L-Systems



APRIL 28, 2016

FH-WEDEL
Timo Schmidt
Inf101010
Prüfungsleistung: PB4

Contents

Lindenmayer-System

L-Systems are invented by Astred Lindenmayer to describe the growing behavior of plants. The growth of plants is based on continual repetition. On cellular level even more than on physiognomy but goal this work is just to visualize natural looking plants based on L-Systems, since it is much easier to generate natural looking constructs than modelling them by hand.

Therefor Lindenmayer-Systems are based on classical mathematical grammars. It starts with an axiom and uses a set of production rules to create a word. The resulting word is used to build a natural looking plant / line-construct. Other than normal grammars production rules for L-Systems should never result in a final word, it should always be possible to replace another variable with one rule.

```
Axiom = F--F--F

Production
Rules:
F = F++G++F
G = +GF-G
```

Listing 1 – Simple grammar

General method

As mentioned before Lindenmayer-Systems are based on mathematical grammars. Listing 1 shows an axiom and a set of production rules. So without any replacements (or zero iterations) the resulting word of that specific system is *F--F--F*. What happens after one and two iterations of replacing variables is shown in Listing 2.

```
Iteration 0:
F--F--F

Iteration 1:
F++G++F--F++G++F--F++G++F

Iteration 2:
F++G++F+++GF-G++F++G++F--F++G++F+++GF-G++F++G++F--F++G++F+++GF-G++F++G++F
```

Listing 2 – Iterating through gammar

What happened? To iterate the axiom once, every occurrence of a variable – in the case of the first iteration the only variable in *F* – gets replaced with its production rule. So *F* gets replaced with *F++G++F* all three times. The same happens at iteration 2 with a slight deviation, now also *G* has to be replaced with *+GF-G*. Of course one can go for another x iterations till we got a really big construct.

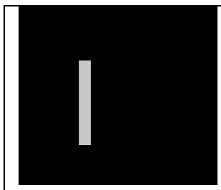
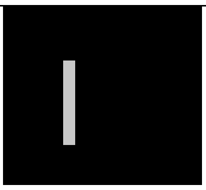
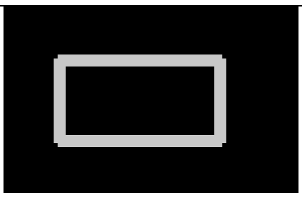
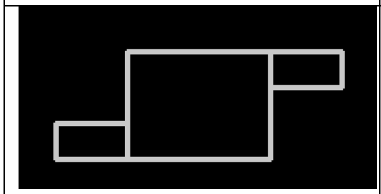
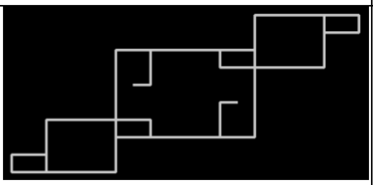
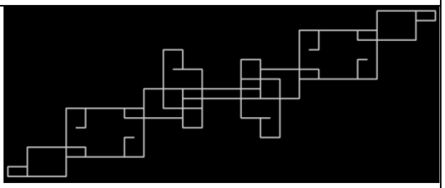
From word to plant

In order to generate a tree or any plant out of the resulting word we need to specify some constants. This implementation uses the following:

-	Turn clockwise about a global angle
+	Turn anti-clockwise about a global angle
[]	Create a new branch
(r,g,b)	To color the following with this color
{w,d}	Specify width (w) and length (d) of the following

Listing 3 – Constants in production rules

So if we use an angle of 90 degrees the results for the first five iterations will look like this:

 Iteration 0	 Iteration 1	 Iteration 2
 Iteration 3	 Iteration 4	 Iteration 5

Listing 4 – Graphical iteration through simple grammar

Branching

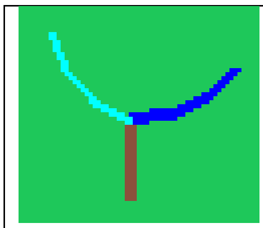
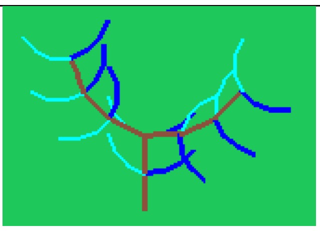
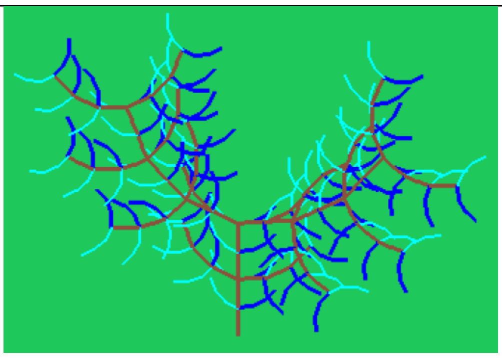
The constructions shown in Listing 4 do look continual but it will never look like a tree or any other plant, at best it could create an alga. So one has to be able to define branching spots. That's what we use those edgy brackets [] for. The opening bracket [defines the beginning of a new branch, whereas the closing bracket] defines the end of a branch. After a branch we will start at the root of the branch again to complete the main branch. The rules defined in Listing 5 create a tree with wide spreading branches.

```
Axiom = F

Angle = 22

Production Rules:
F=FF-[---F+F+F]+[+++F-F-F]
```

Listing 5 – Grammar with branches

 Iteration 1	 Iteration 2	 Iteration 3

Listing 6 – Graphical iteration through grammar with branches

Colorization

To color the branches in different colors simply apply the color to a rule (see Listing 7).

```
F=(140,80,60)FF-[(0,0,255)---F+F+F]+[(0,255,255)+++F-F-F]
```

Listing 7 – Grammar with colorcodes

Width and length

To change the width and length of a specific branch, again the rule gets modified (see Listing 8).

```
F={3,7}(140,80,60)FF-[{3,4}(0,0,255)---F+F+F]+[{2,2}(0,255,255)+++F-F-F]
```

Listing 8 – Grammar with specific width and length of branches

Whereas the first parameter defines the width and the second defines the length.

Examples

To run examples simply use the parameter -e.

```
python lind.py -e  "Cross"
                   "Dragon Curve"
                   "Highway Dragon"
                   "Koch Snowflake"
                   "Koch Curve"
                   "Joined Cross Curves"
                   "Lace"
                   "Pleasant Error"
                   "Sierpinski Median Curve"
                   "Sierpinski Carped"
                   "Sierpinski Triangle 1"
                   "Sierpinski Triangle 2"
                   "Space Filling Curve"
                   "Fractal Plant"
                   "Pond Weed"
                   "Wispy Free"
                   "Tree" (default)
```

Install

To run this program you need to install python 2.7 and pygame. Note: there is no officially compiled version of pygame for x64 platforms. To avoid any issues I suggest to install python and pygame for x86 platforms.

Run

Simply open a terminal and navigate to the folder you saved *lind.py* in.
Start program with default settings: `python lind.py`
And use the following parameters:

`-ax AXIOM:string`	Set axiom to start with
`-prod PRODUCTIONS:string`	Set production rules to iterate axiom
`-s ANGLE:int`	Set angle to start drawing
`-a ANGLE:int`	Set angle to rotate
`-d DISTANCE:int`	Set standard length of one tree part
`-w WIDTH:int`	Set standard width of each tree part
`-r ANGLE:int`	Set maximum random angle to variate Default: 0 Doesn't work properly with -g
`-rec ITERATIONS:int`	Set number of iterations
`-c COLOR:int,int,int`	Set background-color Default: black
`--coord COORD:int,int`	Set coordinate to begin drawing
`--metrics COORD:int,int`	Set metrics of opened window
`-g`	Let the L-System grow each time step till given number of iterations is achieved Doesn't work properly with –r
`--shrink`	Tree gets thinner with each branch
`-e EXAMPLE`	Start one of the given examples. Combinable with other parameters. Note: -ax and -prod will overwrite the axiom and production rules of the example.

Listing 9 – Usable parameters

LINKS

Python 2.7: https://www.python.org/download/releases/2.7/

PyGame: http://www.pygame.org/download.shtml