L-systems

Carlos Andújar

Feb 2014

Aristid Lindenmayer 1925–1989

P. Prusinkiewicz, A. Lindenmayer:
The Algorithmic Beauty of Plants.
Springer (2004)

Example 1

Variables:

F → stem (only grows in length)

X → bud (produces stems & buds)

Rules:

F → FF

X → F[X]X

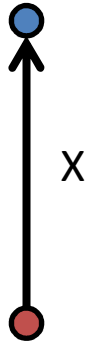
Axiom: X

Interpreter (pos-angle turtle):

F, X = advance turtle (draw segment)

[= push state & rotate -45

] = pop state & rotate +45



N=0 X

Example 1

Variables:

F → stem (only grows in length)

X → bud (produces stems & buds)

Rules:

F → FF

X → F[X]X

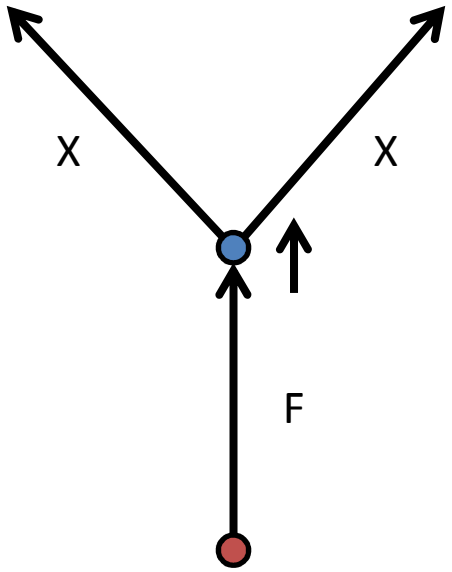
Axiom: X

Interpreter (pos-angle turtle):

F, X = advance turtle (draw segment)

[= push state & rotate -45

] = pop state & rotate +45



N=1 F[X]X

Example 1

Variables:

F → stem (only grows in length)

X → bud (produces stems & buds)

Rules:

F → FF

X → F[X]X

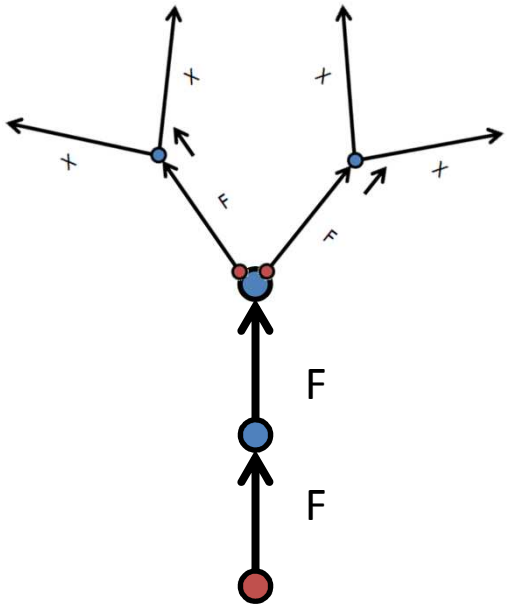
Axiom: X

Interpreter (pos-angle turtle):

F, X = advance turtle (draw segment)

[= push state & rotate -45

] = pop state & rotate +45



N=2 FF[F[X]X]F[X]X

Example 1

Variables:

F → stem (only grows in length)

X → bud (produces stems & buds)

Rules:

F → FF

X → F[X]X

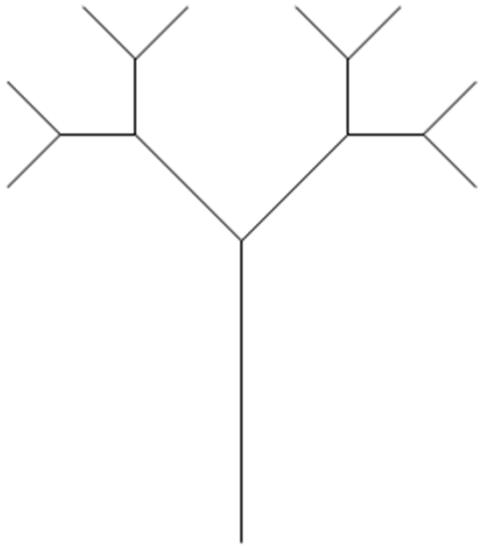
Axiom: X

Interpreter (pos-angle turtle):

F, X = advance turtle (draw segment)

[= push state & rotate -45

] = pop state & rotate +45



N=3 FFFF[FF[F[X]X]F[X]X]FF[F[X]X]F[X]X

Example 1

Variables:

F → stem (only grows in length)

X → bud (produces stems & buds)

Rules:

F → FF

X → F[X]X

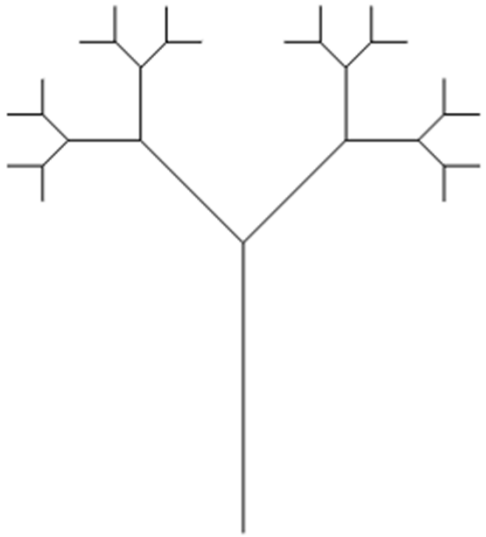
Axiom: X

Interpreter (pos-angle turtle):

F, X = advance turtle (draw segment)

[= push state & rotate -45

] = pop state & rotate +45



N=4

TTTTTTTT[TTTT[TT[T[H]H]T[H
]H]TT[T[H]H]T[H]H]TTTT[TT[T
[H]H]T[H]H]TT[T[H]H]T[H]H

Example 1

Variables:

F → stem (only grows in length)

X → bud (produces stems & buds)

Rules:

F → FF

X → F[X]X

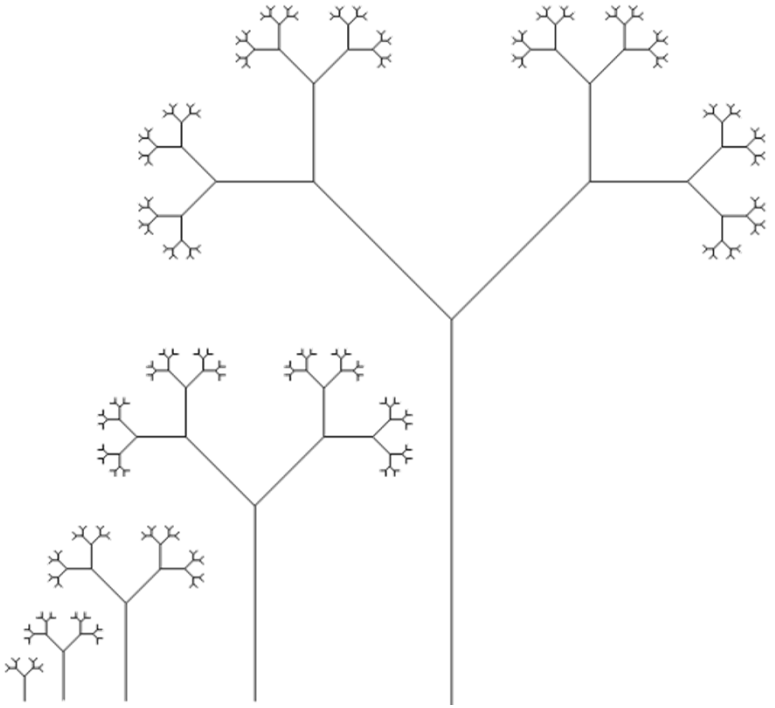
Axiom: X

Interpreter (pos-angle turtle):

F, X = advance turtle (draw segment)

[= push state & rotate -45

] = pop state & rotate +45



N=3,4,5,6 & 7

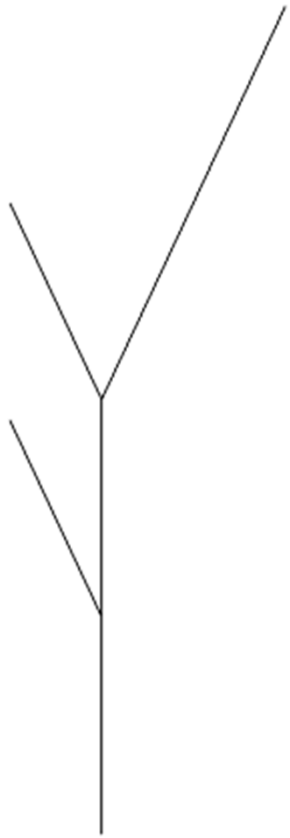
Example 2

Variables:
- F → stem (only grows in length)
- X → bud (produces stems & buds)

Rules:
- F → FF
- X → F-[[X]+X]+F[-X]+FX

Axiom: X

Interpreter (pos-angle turtle):
- F, X = advance turtle (draw segment)
- [= push state
-] = pop state
- + = rotate +25
- - = rotate -25



N=1 F-[[X]+X]+F[-X]+FX

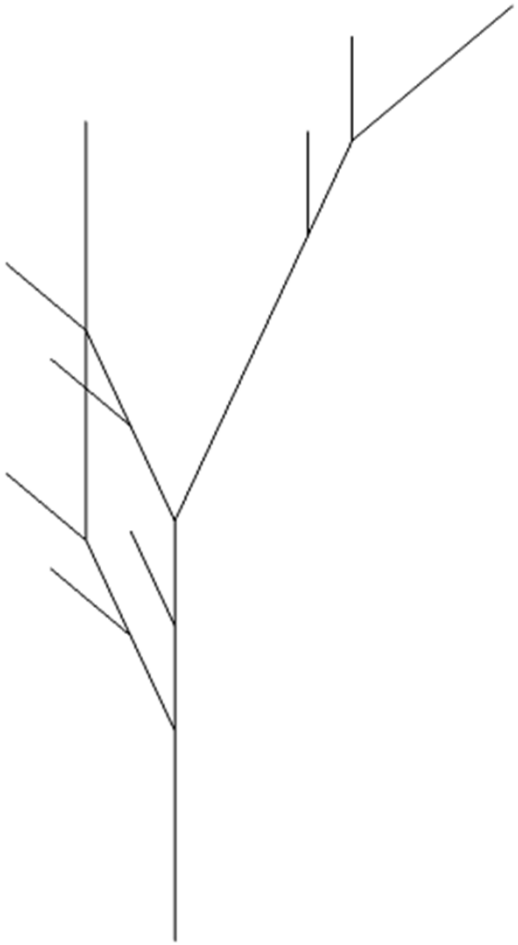
Example 2

Variables:
- F → stem (only grows in length)
- X → bud (produces stems & buds)

Rules:
- F → FF
- X → F-[[X]+X]+F[-X]+FX

Axiom: X

Interpreter (pos-angle turtle):
- F, X = advance turtle (draw segment)
- [= push state
-] = pop state
- + = rotate +25
- - = rotate -25



N=2

FF-[[F-[[X]+X]+F[-X]+FX]+F-[[X]+X]+F[-X]+FX]+FF[-F-[[X]+X]+F[-X]+FX]+FFF-[[X]+X]+F[-X]+FX

Example 2

Variables:
- F → stem (only grows in length)
- X → bud (produces stems & buds)

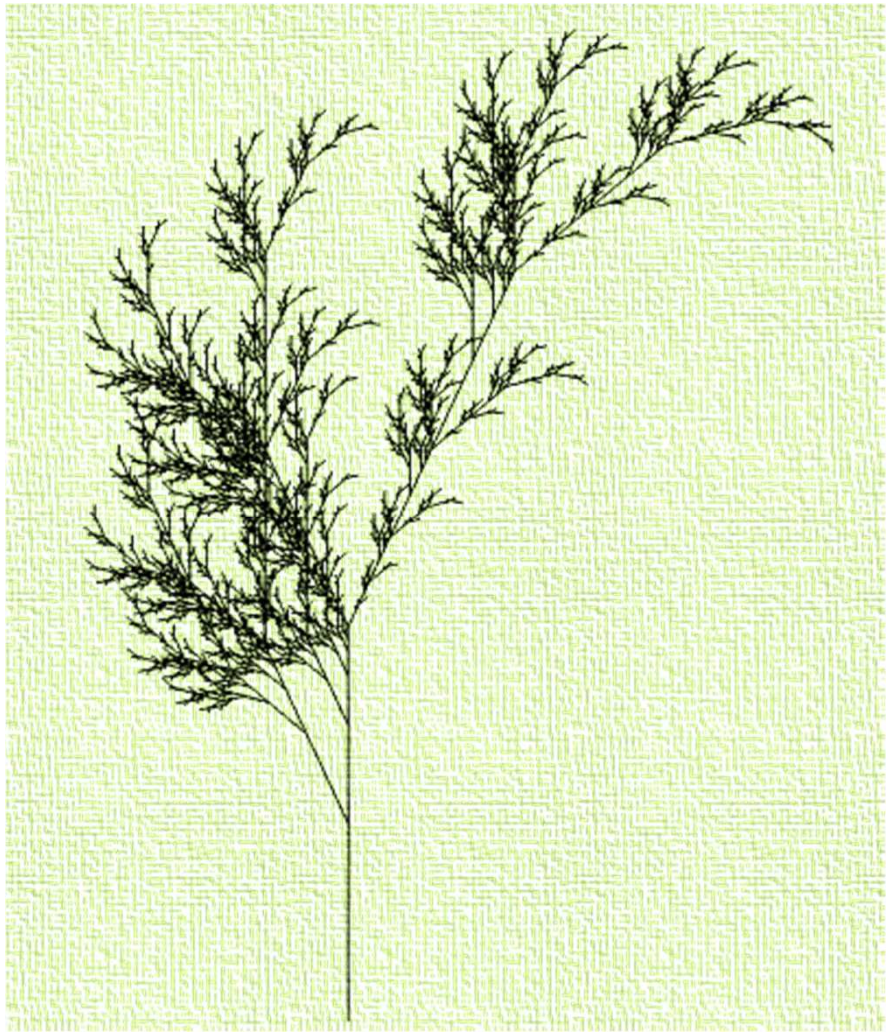
Rules:
- F → FF
- X → F-[[X]+X]+F[-X]+FX

Axiom: X

Interpreter (pos-angle turtle):
- F, X = advance turtle (draw segment)
- [= push state
-] = pop state
- + = rotate +25
- - = rotate -25

N=6

Example 2

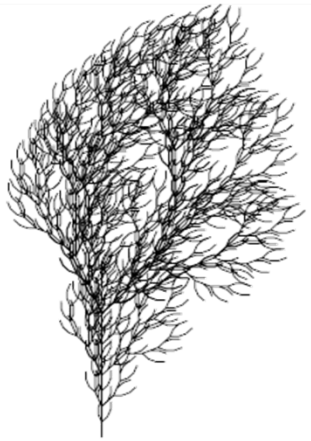


Example 2 - variations

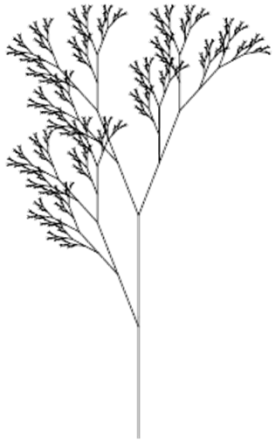
Other plant-like branching structures from the CPFG Manual (Prusinkiewicz's group)

F ⟶ F[+F]F[-F]F

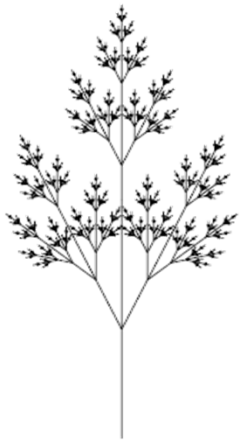
F ⟶ F[+F]F[-F][F]



F ⟶ FF-[-F+F+F]+[+F-F-F]



X ⟶ F[+X]F[-X]+X
F ⟶ FF



X ⟶ F[+X][-X]FX
F ⟶ FF

X ⟶ F-[[X]+X]+F[+FX]-X
F ⟶ FF

Example 3 – stochastic grammar

Rules:

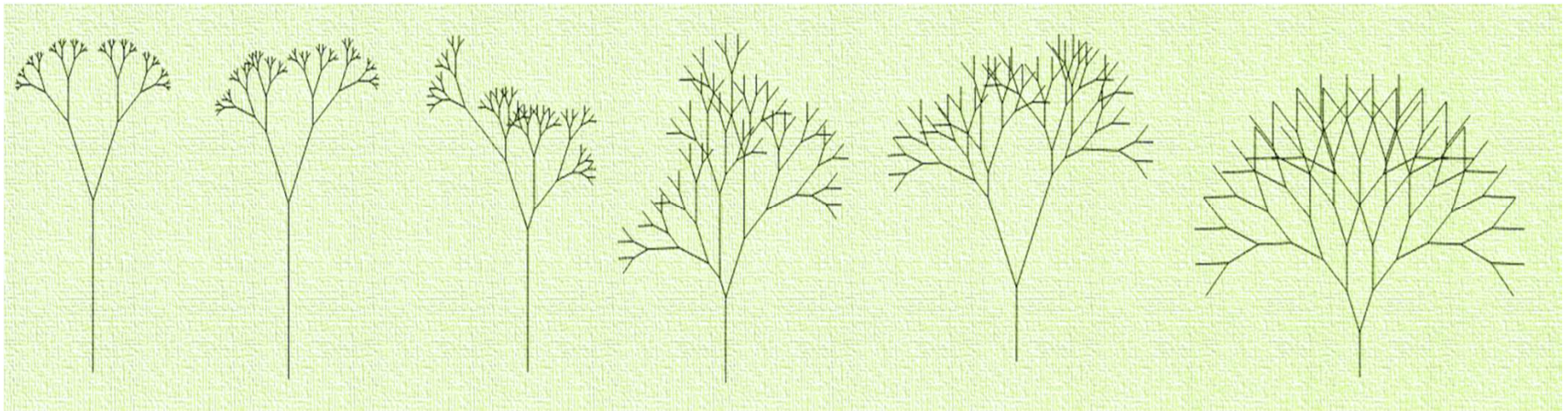
F (p) → FF

F (1-p) → F

X → F[X]X

N=6, ∂=25˚

p=1.0 p=0.8 p=0.6 p=0.4 p=0.2 p=0.0



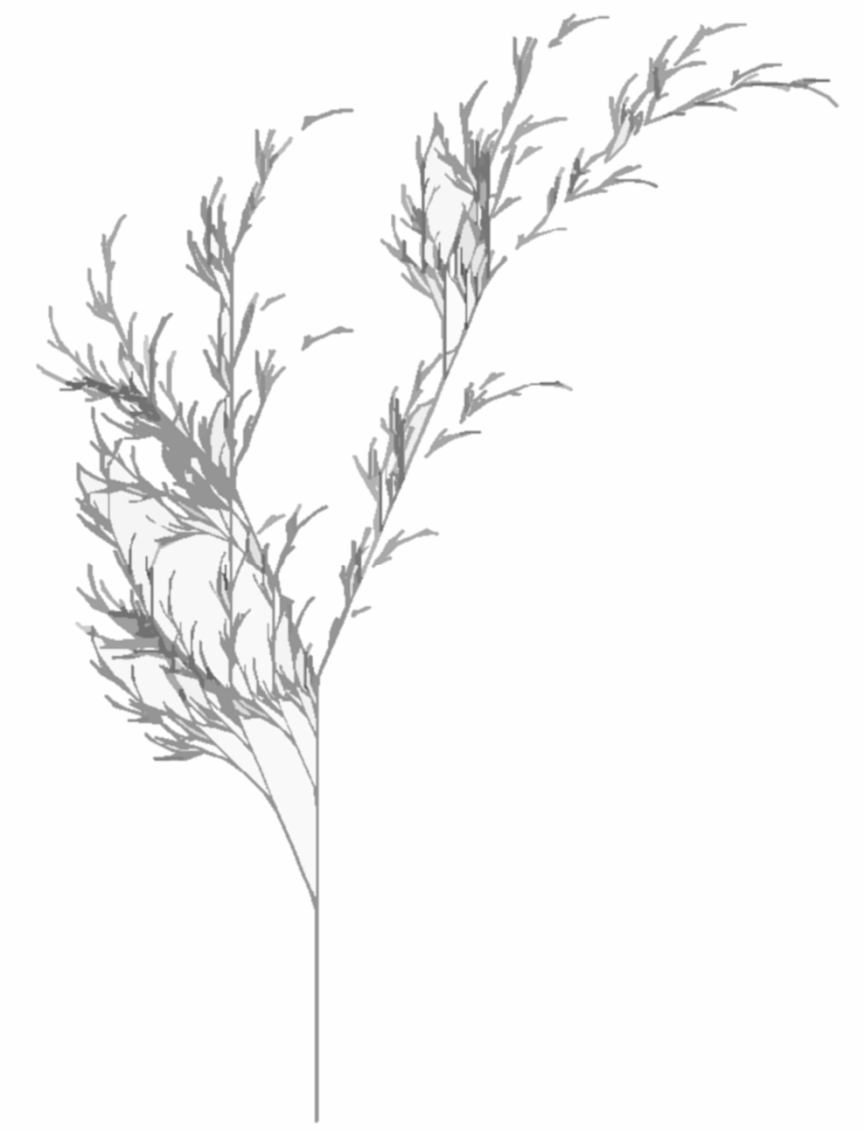
Example 4 – stochastic system

Rules:

F → FF

X (0.4) → F-[[X]+X]+F[-X]+FX

X (0.6) → F-[[X]+X]+F+FX



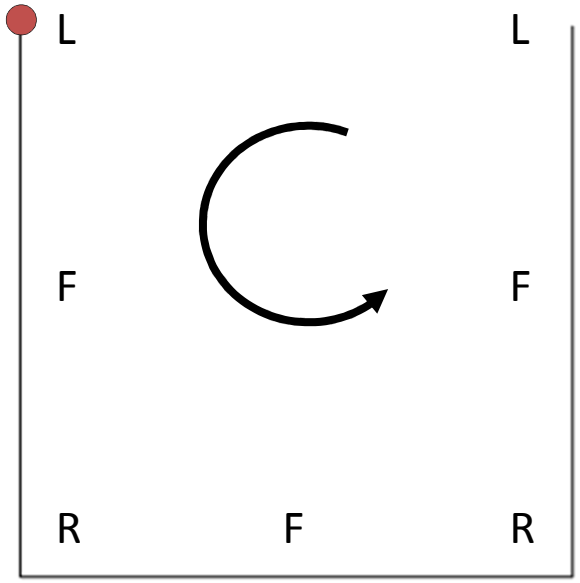
Example 5 – Hilbert curve

Rules:

L → +RF-LFL-FR+

R → -LF+RFR+FL-

Start: R



R -LF+RFR+FL-

n=0 n=1

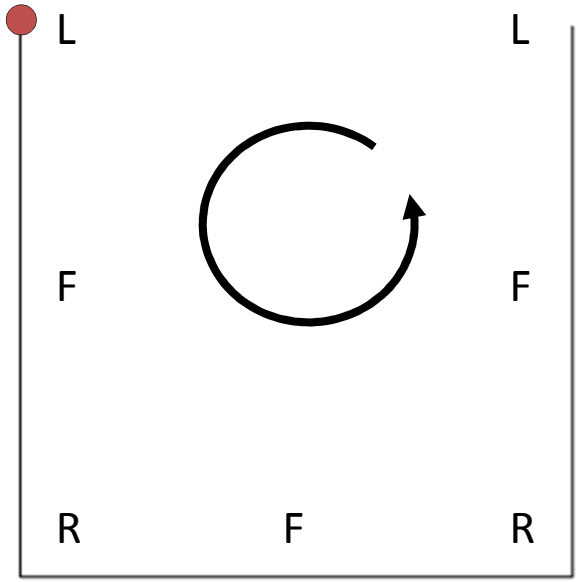
Example 5 – Hilbert curve

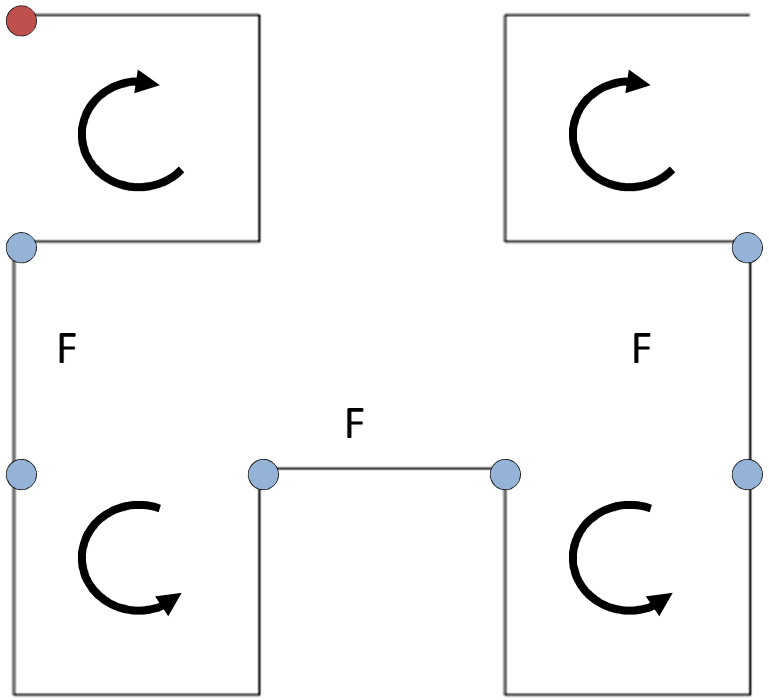
Rules:

L → +RF-LFL-FR+

R → -LF+RFR+FL-

Start: R

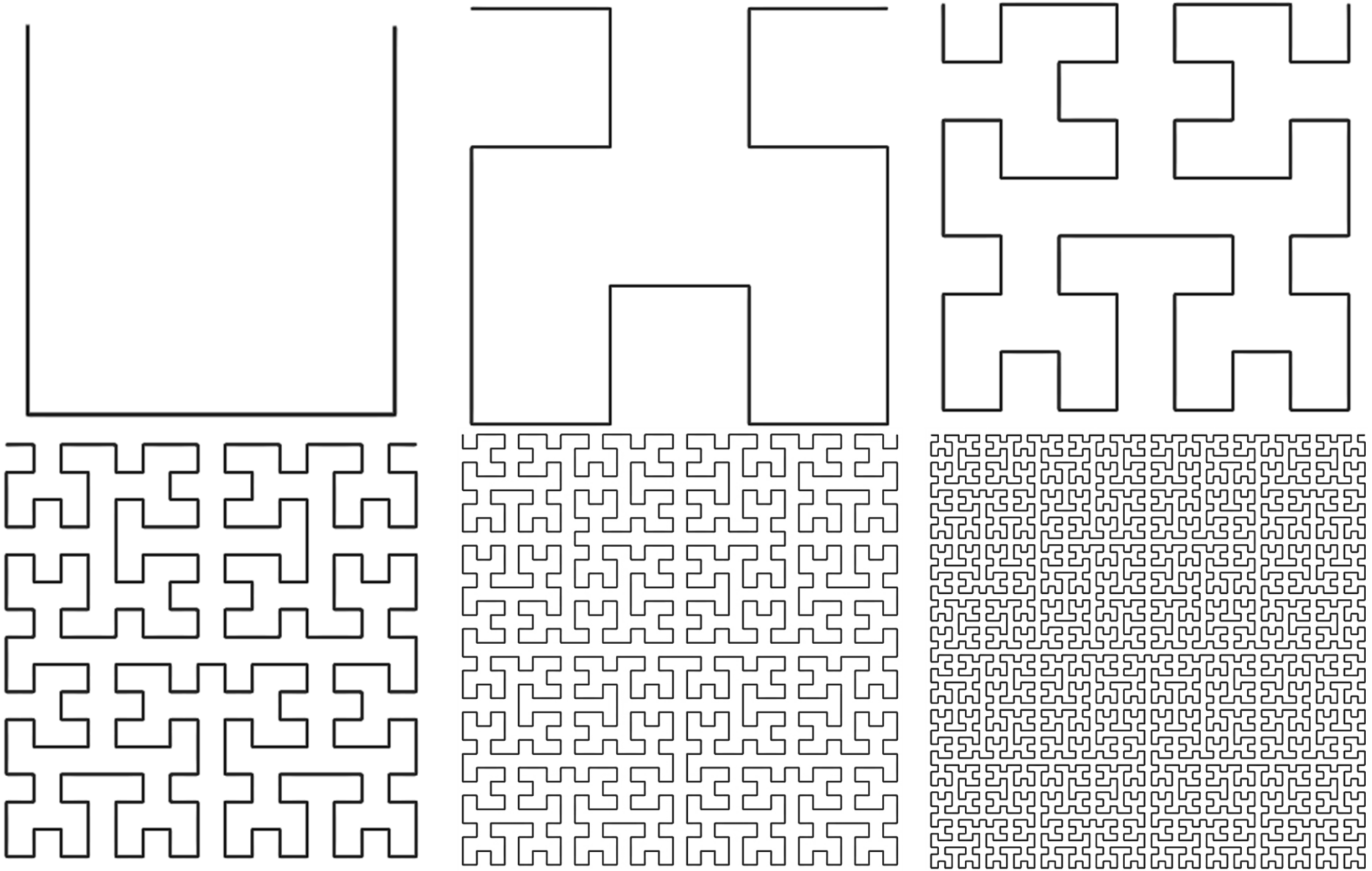




R -LF+RFR+FL- -+RF-LFL-FR+F+-LF+RFR+FL-F-LF+RFR+FL-+F+RF-LFL-FR+-

n=0 n=1 n=2

Example 5 – Hilbert curve



Example 6 – parametric grammar

Rules (in L-Py language)

```
phi = 90
alpha = 32
beta  = 20

Axiom: A(1)
derivation length: 100
production:

A(k) :
    if random.random() <= 0.2:
        produce /(phi)[+(alpha)C(k)]-(beta)C(k)  # two branches
    else:
        produce /(phi)[+(alpha);~l(1.2)]-(beta)C(k) # one leaf & one branch

C(k) :
    if k < 18:
        produce F(1,0.05+0.25/k)A(k+1)  # advance and continue recursion
    else:
        produce F(1,0.05+0.25/k)        # advance and finish recursion
```

Example 6 – parametric grammar

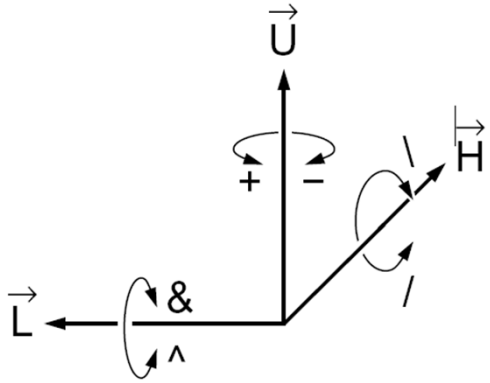
Rules (in L-Py language)

Recursive calls (variables)

```
phi = 90
alpha = 32
beta  = 20

Axiom: A(1)
derivation length: 100
production:

A(k) :
    if random.random() <= 0.2:
        produce /(phi)[+(alpha)C(k)]-(beta)C(k)  # two branches
    else:
        produce /(phi)[+(alpha);~l(1.2)]-(beta)C(k) # one leaf & one branch

C(k) :
    if k < 18:
        produce F(1,0.05+0.25/k)A(k+1)  # advance and continue recursion
    else:
        produce F(1,0.05+0.25/k)        # advance and finish recursion
```

Example 6 – parametric grammar

Rules (in L-Py language)



/(phi) = roll left around heading vector

```
phi = 90
alpha = 32

production:

A(k) :
    if random.random() <= 0.2:
        produce /(phi)[+(alpha)C(k)]-(beta)C(k)  # two branches
    else:
        produce /(phi)[+(alpha);~l(1.2)]-(beta)C(k) # one leaf & one branch

C(k) :
    if k < 18:
        produce F(1,0.05+0.25/k)A(k+1)  # advance and continue recursion
    else:
        produce F(1,0.05+0.25/k)        # advance and finish recursion
```

Example 6 – parametric grammar

Rules (in L-Py language)

```
phi = 90
alpha = 32
beta  = 20

Axiom: A(1)
derivation length: 100
production:

A(k) :
    if random.random() <= 0.2:
        produce /(phi)[+(alpha)C(k)]-(beta)C(k)  # two branches
    else:
        produce /(phi)[+(alpha);~l(1.2)]-(beta)C(k) # one leaf & one branch

C(k) :
    if k < 18:
        produce F(1,0.05+0.25/k)A(k+1)  # advance and continue recursion
    else:
        produce F(1,0.05+0.25/k)        # advance and finish recursion
```

+(alpha) = turn left around up vector

Example 6 – parametric grammar

Rules (in L-Py language)

```
phi = 90
alpha = 32
beta  = 20

Axiom: A(1)
derivation length: 100
production:

A(k) :
    if random.random() <= 0.2:
        produce /(phi)[+(alpha)C(k)]/(beta)C(k)  # two branches
    else:
        produce /(phi)[+(alpha);~l(1.2)]-(beta)C(k) # one leaf & one branch

C(k) :
    if k < 18:
        produce F(1,0.05+0.25/k)A(k+1)  # advance and continue recursion
    else:
        produce F(1,0.05+0.25/k)        # advance and finish recursion
```

~l(size) = draw a leaf

Example 6 – parametric grammar

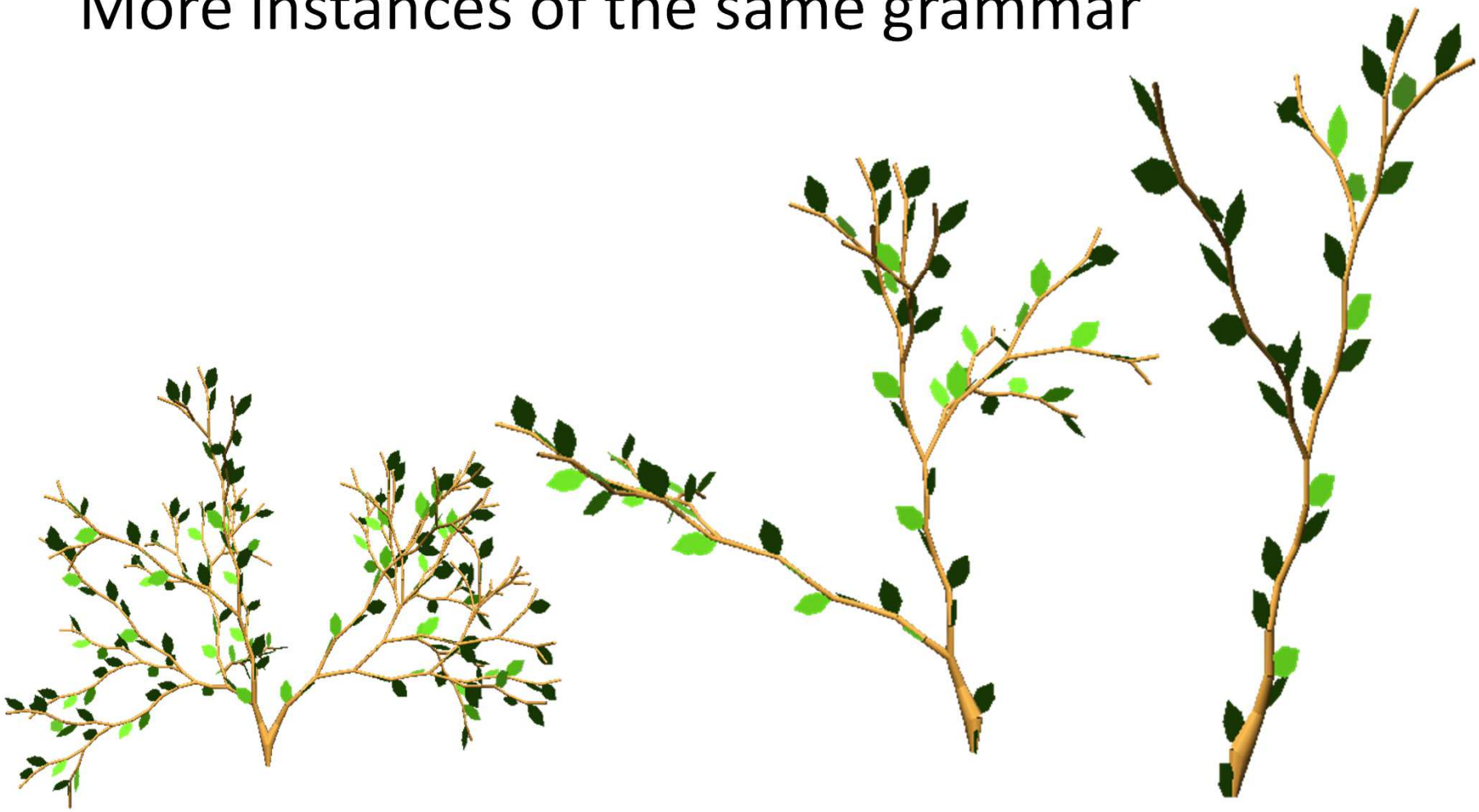
Rules (in L-Py language)

```
phi = 90
alpha = 32
beta  = 20

Axiom: A(1)
derivation length: 100
production:

A(k) :
    if random.random() <= 0.2:
        produce /(phi)[+(alpha)[&(1.2)]-(beta)C(k)  # two branches
    else:
        produce /(phi)[+(alpha)[&(1.2)]-(beta)C(k) # one leaf & one branch

C(k) :
    if k < 18:
        produce F(1,0.05+0.25/k)A(k+1)  # advance and continue recursion
    else:
        produce F(1,0.05+0.25/k)        # advance and finish recursion
```

F(length, radius) = draw a cylinder

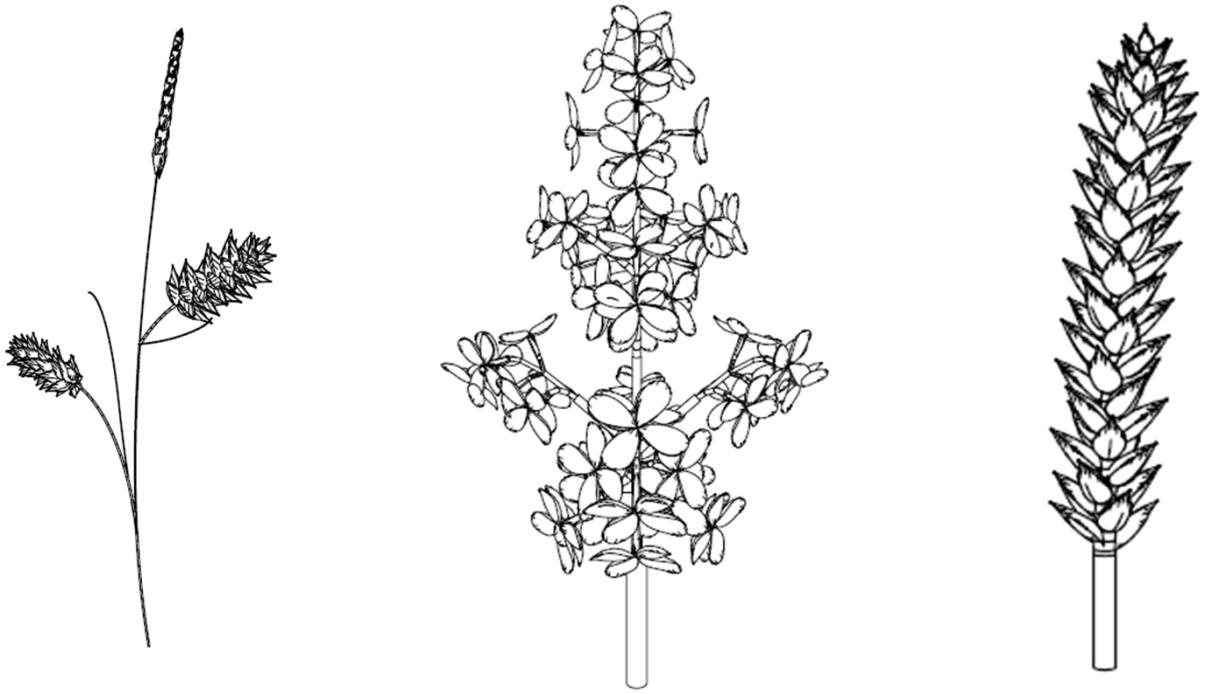
Example 6 – parametric grammar

More instances of the same grammar

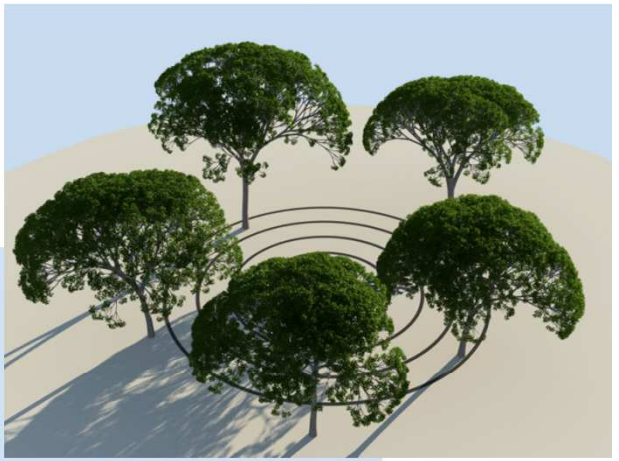


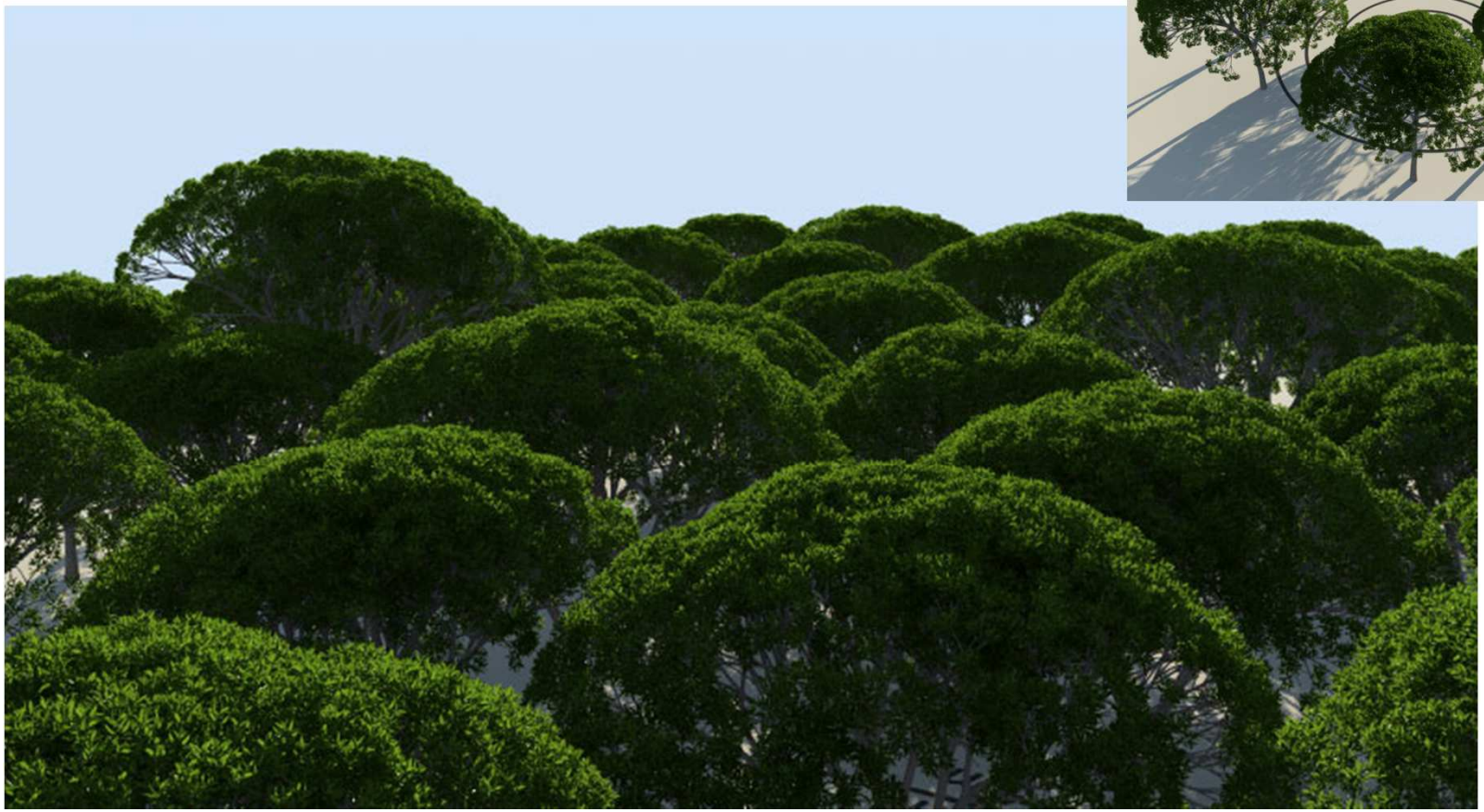
More examples

See grammars in the CPFG User's Manual



Further examples





Source: www.3dmentor.ru

Links

- P. Prusinkiewicz, A. Lindenmayer: The Algorithmic Beauty of Plants, Springer (2004) http://algorithmicbotany.org/papers/abop/abop.pdf
- Biological Modeling and Visualization research group @ Univ Calgary (Prusinkiewicz's group) http://algorithmicbotany.org
- Virtual Plants team @ INRIA & L-Py: http://openalea.gforge.inria.fr/