Projet
L-Systèmes

Stéphane Vialette

LIGM, Université Paris-Est Marne-la-Vallée

30 octobre 2009

1 Introduction

2 Définitions

3 Tortue

4 Python

Outline

1 Introduction

2 Définitions

3 Tortue

4 Python

L-Système (Wikipedia)

Introduction

- Un *L-Système* (ou système de Lindenmayer) est une grammaire formelle, permettant un procédé algorithmique, inventé en 1968 par le biologiste hongrois Aristid Lindenmayer qui consiste à modéliser le processus de développement et de prolifération de plantes ou de bactéries.
- Basée sur une forme récursive de grammaire générative, cette grammaire a été approfondie et mise en œuvre graphiquement par Przemyslaw Prusinkiewicz dans les années 1980.
- Au départ, Lindenmayer avait pensé ce système comme un langage formel qui permettait de décrire le développement d'organismes multicellulaires simples. À cette époque il travaillait sur les levures, les champignons et des algues. Mais l'informatique a permis d'exploiter ce système pour générer graphiquement des calculs de plantes très complexes.

L-Système (Wikipedia)

Introduction

- Un L-Système est un ensemble de règles et de symboles qui modélisent un processus de croissance d'êtres vivants comme des plantes ou des cellules. Le concept central des L-Systems est la notion de réécriture. La réécriture est une technique pour construire des objets complexes en remplaçant des parties d'un objet initial simple en utilisant des règles de réécriture.
- Pour ce faire, les cellules sont modélisées à l'aide de symboles. À chaque génération, les cellules se divisent, i.e. un symbole est remplacé par un ou plusieurs autres symboles formant un mot.

Illustration (Wikipedia)

Outline

1 Introduction

2 Définitions

3 Tortue

4 Python

Définitions

L-Système

Un L-Système est une grammaire formelle qui comprend :

1. Un alphabet V : l'ensemble des variables du L-Système. V^* est l'ensemble des mots que l'on peut construire avec les symboles de V, et V^+ l'ensemble des mots contenant au moins un symbole.
2. Un ensemble de valeur constantes S. Certains de ces symboles sont communs à tous les L-Systèmes.
3. Un axiome de départ ω choisi parmi V^+ , c'est-à-dire l'état initial.
4. Un ensemble de règles, noté P, de reproduction des symboles de V.

Un L-Système est alors noté (V, S, ω, P).

L'algue de Lindenmayer

Définition

- Alphabet : $V = \{A, B\}$,
- Constantes : $S = \emptyset$,
- Axiome : $\omega = A$,
- Régles : $P = \{(A \rightarrow AB), (B \rightarrow A)\}$.

Notation

Algue
{
Axiom A
A=AB
B=A
}

L'algue de Lindenmayer

Example

1. A
2. AB
3. AB A
4. AB A AB
5. AB A AB AB A
6. AB A AB AB A AB A AB
7. AB A AB AB A AB A AB AB A AB AB A

La suite de Fibonacci

Définition

- Alphabet : $V = \{A, B\}$,
- Constantes : $S = \emptyset$,
- Axiome : $\omega = A$,
- Régles : $P = \{(A \rightarrow B), (B \rightarrow BA)\}$.

Notation

Algue
{
Axiom A
A=AB
B=A
}

La suite de Fibonacci

Example

1. A
2. B
3. AB
4. B AB
5. AB B AB
6. B AB AB B AB
7. AB B AB B AB AB B AB

Système stochastique

Notation

Plante_Stochastique
{
angle 20
axiom X
X=(0.2)F[++X]F[−X]+X
X=(0.8)F[+X]F[−X]+X
F=(1.0)FF
}

Système stochastique

Exemple 1

- $n = 0$: `X`
- $n = 1$: `F[++X]F[−X]+X`
- $n = 2$:
  `FF[++F[+X]F[−X]+X]FF[−F[+X]F[−X]+X]+F[+X]F[−X]+X`

Exemple 2

- $n = 0$: `X`
- $n = 1$: `F[+X]F[−X]+X`
- $n = 2$:
  `F[+F[++X]F[−X]+X]F[−F[++X]F[−X]+X]+F[++X]F[−X]+X`

Outline

1 Introduction

2 Définitions

3 Tortue

4 Python

Tortue

Description

- Imaginons que nous avons un crayon à la main et qu'elle se balade sur la feuille sous nos ordres : "monte d'un cran, puis tourne de de 20° à gauche, déplace toi deux fois de un cran, mémorise ta position et avance encore d'un cran, lève-toi puis repose-toi sur la position mémorisée" et ainsi de suite...
- Il a donc fallu inventer des symboles variants $\in V$, ou constants $\in S$, pour permettre de guider cette main. Plusieurs d'entre eux ont été normalisés, ils font partie de ce qu'on appelle la "*Turtle interpretation*".

Tortue

Symboles

- `F` : Se déplacer d'un pas unitaire ($\in V$)
- `+` : Tourner à gauche d'angle α ($\in S$)
- `-` : Tourner à droite d'un angle α ($\in S$)
- `&` : Pivoter vers le bas d'un angle α ($\in S$)
- `^` : Pivoter vers le haut d'un angle α ($\in S$)
- `<` : Roulez vers la gauche d'un angle α ($\in S$)
- `>` : Roulez vers la droite d'un angle α ($\in S$)
- `|` : Tourner sur soi-même de 180° ($\in S$)
- `[` : Sauvegarder la position courante ($\in S$)
- `]` : Restaurer la dernière position sauvée ($\in S$)

La courbe de Koch

Définition

- Alphabet : $V = \{F\}$,
- Constantes : $S = \{+, -\}$,
- Axiome : $\omega = F$,
- Régles : $P = \{(F \rightarrow F + F - F - F + F)\}$.

Notation

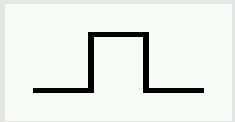
Courbe_de_Koch
{
angle 90
axiom F
F=F+F−F−F+F
}

La courbe de Koch

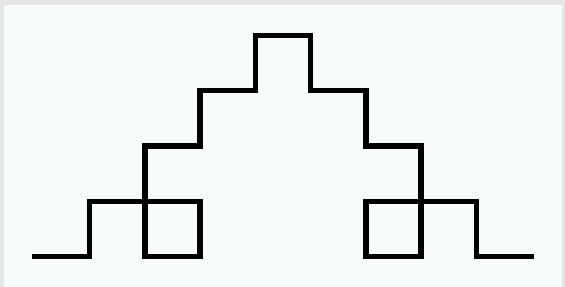
Exemple

- $n = 0$: `F`

- $n = 1$: `F+F-F-F+F`



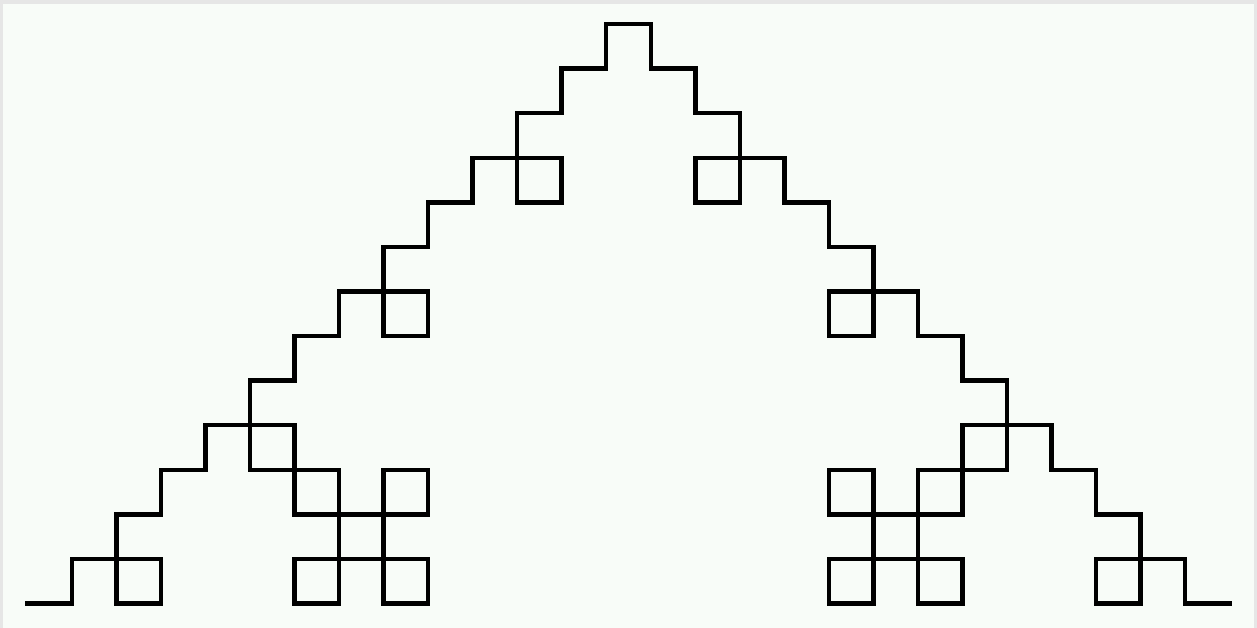
- $n = 2$: `F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F`



La courbe de Koch

Exemple

- $n = 3$: `F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F+`
  `F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F-`
  `F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F-`
  `F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F+`
  `F+F-F-F+F+F+F-F-F+F-F+F-F-F+F-F+F-F-F+F+F+F-F-F+F`



Sauvegarde de la position courante

Définition

- Alphabet : $V = \{F, X\}$,
- Constantes : $S = \{+, -, [,]\}$,
- Axiome : $\omega = X$,
- Régles : $P = \{(X \rightarrow F[+X]F[-X] + X)), (F \rightarrow FF)\}$.

Notation

Plante
{
angle 20
axiom X
X=F[+X]F[−X]+X
F=FF
}

Outline

1 Introduction

2 Définitions

3 Tortue

4 Python

Session

Example

```
from lsystem import LSystem

plante = LSystem('plante.lsyst')
print type(plante)
print 'name :', plante.name()
print 'axiom :', plante.axiom()
print 'rules :', plante.rules()
print 'angle :', plante.angle()
print 'lsyst :'
print plante
```

Session

Output

```
$ python session1.py
class '__main__.LSystem'>
name : Plante
axiom : X
rules : {X : F[+X]F[-X]+X,F : FF}
angle : 20
lsyst :
Plante
{
angle 20
axiom X
rules {X : F[+X]F[-X]+X,F : FF}
}
$
```

LSystem

Accès aux éléments

- `LSystem(filename)` : Création d'une instance `Lsystem` correspondant au L-système donné dans le fichier `filename`.
- `variables()` : Retourne la listes des variables du L-système.
- `constants()` : Retourne la listes des constantes du L-système.
- `name()` : Retourne le nom du L-système.
- `axiom()` : Retourne l'axiome du L-système.
- `angle()` : Retourne l'angle du L-système.
- `rules()` : Retourne les rèles du L-système.

LSystem

Exceptions

- `LSystemParseError` : Exception levée si le L-système est mal-formé.
- `LSystemError` : Exception propre au L-système :
  - l'axiome n'apparaît pas comme prémisse d'une règle,
  - une variable apparaît plusieurs fois en prémisse d'une régle,
  - ...

Session

Example

```
from lsystem import LSystem
from lsystem import LSystemTurtle
from lsystem import LSystemTurtleCodeFactory

# create a l-system object
plante = LSystem('plante.lsyst')
# compute some iterations
states = plante.iterates(4)
# get the turtle code and do something with it
turtle_code = LSystemTurtleCodeFactory(states)
turtle_code.exec()
turtle_code.write('some_filename')
# or directly draw the iterations
LSystemTurtle(states)
```

Liens

Modules

- **turtle** : tortue graphique.
  `http ://docs.python.org/library/turtle.html`
- **ply** : Lex/Yacc avec Python.
  `http ://www.dabeaz.com/ply/`
- **unittest** : tests unitaires.
  `http ://docs.python.org/library/unittest.html`
- **pylint** : analyseur de code.
  `http ://www.logilab.org/857`