Projet de XML - Les L-Systèmes

V. Padovani, P. Letouzey, J. Chroboczek

Un *système de Lindenmayer* (L-système) est un système de réécriture permettant de générer des graphiques modélisant la croissance des plantes. Le but de la première partie de ce projet est de convertir par programme, au format XML, le contenu d'un fichier de données brutes décrivant un ensemble de L-systèmes. Celui de la seconde est de générer à l'aide de XSLT et à partir du XML produit, des graphiques engendrés par ces L-systèmes au format SVG.

I) Les L-systèmes

Les définitions ci-dessous sont purement formelles, et ne prendront véritablement sens qu'à la section suivante. Un système de Lindenmayer n'est rien de plus qu'un mécanisme de production de suites de symboles : une suite initiale, une règle permettant de transformer une suite produite en une nouvelle suite par substitution de ses symboles.

1. Étant donné un ensemble de symboles $\mathcal{S}$, une *chaîne parenthésée* sur $\mathcal{S}$ est une chaîne d'éléments de $\mathcal{S}$ dont certaines sous-chaînes sont encadrées par des crochets « [» et «] ». Par exemple,
$$\mathrm{F[PF][F[PF]MF]MFF]}$$
est une chaîne parenthésée sur l'ensemble $\mathcal{S} = \{\mathrm{F}, \mathrm{P}, \mathrm{M}\}$.

2. Une *substitution* sur un ensemble de symboles $\mathcal{S}$ est une fonction associant à des éléments de $\mathcal{S}$ des chaînes parenthésées sur $\mathcal{S}$. Par exemple, pour $\mathcal{S} = \{\mathrm{F}, \mathrm{P}, \mathrm{M}\}$, la fonction
$$\begin{cases} \mathrm{F} & \mapsto & \mathrm{F[FM]} \\ \mathrm{M} & \mapsto & \mathrm{[FP]} \end{cases}$$
est une substitution sur $\mathcal{S}$.

3. *Appliquer* une substitution à une chaîne parenthésée consiste à parcourir cette chaîne en remplaçant chaque symbole dans le domaine de cette substitution par son image. Par exemple, en appliquant la substitution précédente à FPM, on obtient la chaîne F[FM]P[FP].

4. Un *L-système* est la donnée d'un triplet $\mathcal{L} = (\mathcal{S}, w, \sigma)$ où :
   - $\mathcal{S}$ est un ensemble (fini) de symboles,
   - w est une chaîne parenthésée sur $\mathcal{S}$, appelée *axiome*,
   - σ est une substitution sur $\mathcal{S}$.

5. Pour chaque entier positif n, l'*itération de rang* n de $\mathcal{L} = (\mathcal{S}, w, \sigma)$ est définie par :
   - l'itération de rang 0 est l'axiome w,
   - l'itération de rang $n+1$ est la chaîne obtenue en appliquant σ à l'itération de rang n.

II) Interprétation graphique des L-systèmes

Les symboles d'une chaîne parenthésée peuvent être interprétés comme des opérations graphiques, produisant souvent – mais pas toujours – un tracé de forme arborescente, rappelant la structure d'une plante. La suite de chaînes produite par les itérations d'un L-système rappelle parfois quant à elle la croissance naturelle de ces plantes – l'intention de Lindenmayer était précisément de modéliser ce phénomène.

a) La Tortue

La *Tortue* est un robot virtuel muni d'un crayon et se déplaçant sur un plan. La Tortue peut effectuer trois sortes d'actions :

- pivoter sur place d'un certain angle négatif ou positif,
- avancer d'une certaine longueur, crayon levé,
- avancer d'une certaine longueur, crayon abaissé.

Plus formellement, l'état courant de la Tortue est un triplet (x, y, θ) où (x, y) sont les coordonnées de la Tortue dans le plan et θ est l'angle (en degrés) formé par sa direction de déplacement et l'axe des x.

Un *script* pour la Tortue est une suite d'instructions parmi les cinq ci-dessous. Les trois premières correspondent aux trois sortes d'actions décrites plus haut. Deux instructions supplémentaires permettent de sauvegarder et restaurer l'état de la Tortue :

- `TURN` a : ajouter a degrés à θ,
- `MOVE` n : avancer de n unités,
- `LINE` n : avancer de n unités, en traçant un segment,
- `STORE` : mémoriser l'état courant,
- `RESTORE` : restaurer le dernier état mémorisé non encore restauré.

Par exemple, le script suivant tracera un triangle équilatéral de côté 10 :

```
LINE 10 TURN 120 LINE 10 TURN 120 LINE 10
```

b) Le Traceur

Le *Traceur* est un autre robot virtuel de fonctionnement encore plus primitif. Son état courant est le simple couple de ses coordonnées dans le plan. Le Traceur se déplaçe toujours en ligne droite et peut, à partir de sa position courante, effectuer deux sortes d'actions :

- aller en (x, y), crayon levé,
- aller en (x, y), crayon abaissé.

Un script pour le traceur est une suite d'instructions parmi les deux sortes d'actions précédentes :

- `MOVETO` (x, y) : aller en (x, y), crayon levé,
- `LINETO` (x, y) : aller en (x, y), crayon abaissé.

Il est clair que tout script de la Tortue peut être converti en un script pour le Traceur produisant, aux approximations près, le même rendu graphique. À l'approximation d'une ordonnée près, le script suivant tracera lui aussi un triangle équilatéral de côté 10 :

```
LINETO (10, 0) LINETO (5, 8.66025404) LINETO (0, 0)
```

c) Interprétation des chaînes

Une *interprétation* d'un ensemble de symboles $\mathcal{S}$ est la donnée, pour chaque symbole de $\mathcal{S}$, d'une instruction pour la Tortue [1]. Cela peut-être, par exemple, pour $\mathcal{S} = \{\mathrm{F}, \mathrm{P}, \mathrm{M}\}$:

$$\begin{cases} \mathrm{F} & \mapsto & \texttt{LINE 10} \\ \mathrm{P} & \mapsto & \texttt{TURN 45} \\ \mathrm{M} & \mapsto & \texttt{TURN -45} \end{cases}$$

L'*interprétation d'une chaîne* sur $\mathcal{S}$ relativement à une interprétation de $\mathcal{S}$ est définie par le script obtenu en parcourant cette chaîne et en transformant :

- chaque symbole de $\mathcal{S}$ en son interprétation,
- le symbole « [» par `STORE`,
- le symbole «] » par `RESTORE`.

1. On pourrait aussi associer à chaque symbole un script quelconque.

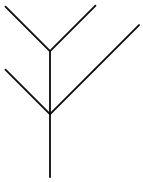

FIGURE 1 – F[PF][F[PF]MF]MFF

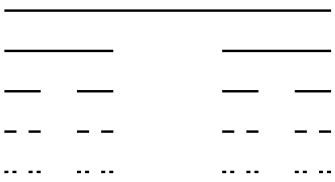

FIGURE 2 – Ensemble de Cantor

Par exemple, avec l'interprétation précédente, la chaîne F[PF][F[PF]MF]MFF produira l'arbre de la Figure 1

d) Évolution de l'interprétation des chaînes des L-systèmes

Exemple : L'ensemble de Cantor

$$G = (\{\mathrm{A}, \mathrm{B}\}, \mathrm{A}, (\mathrm{A} \mapsto \mathrm{ABA}, \mathrm{B} \mapsto \mathrm{BBB})).$$

Ce système produit par itération les chaînes suivantes :

1. A
2. ABA
3. ABABBBABA
4. ABABBBABABBBBBBBBBABABBBABA
5. $\cdots$

Si l'on interprète le symbole A par `LINE 10` et le symbole B par `MOVE 10`, les interprétations des chaînes produites par ce L-système décrivent la construction de l'*ensemble de Cantor* (Figure 2).

Exemple : Courbe de von Koch

$$G = (\{\mathrm{A}, \mathrm{P}, \mathrm{M}\}, \mathrm{APPAPPA}, (\mathrm{A} \mapsto \mathrm{AMAPPAMA})).$$

Ce système produit les chaînes suivantes :

1. APPAPPA
2. AMAPPAMAPPAMAPPAMAPPAMA
3. $\cdots$

En interprétant A comme `LINE 10`, P comme `TURN 60` et M comme `TURN -60`, les interprétations décrivent la construction de la *courbe de von Koch* (Figure 3).

Exemple : Une plante

$$G = (\{\mathrm{A}, \mathrm{P}, \mathrm{M}\}, \mathrm{A}, (\mathrm{A} \mapsto \mathrm{A[PA]A[MA]A})).$$

Ce système produit :

1. A
2. A[PA]A[MA]A
3. A[PA]A[MA]A[PA[PA]A[MA]A]A[PA]A[MA]A[MA[PA]A[MA]A]A[PA]A[MA]A
4. $\cdots$

Si l'on interprète A comme `LINE 10`, P comme `TURN 25`, et M commme `TURN -25`, l'interprétation de chaînes construit, par raffinements successifs, la « plante » de la figure 4.

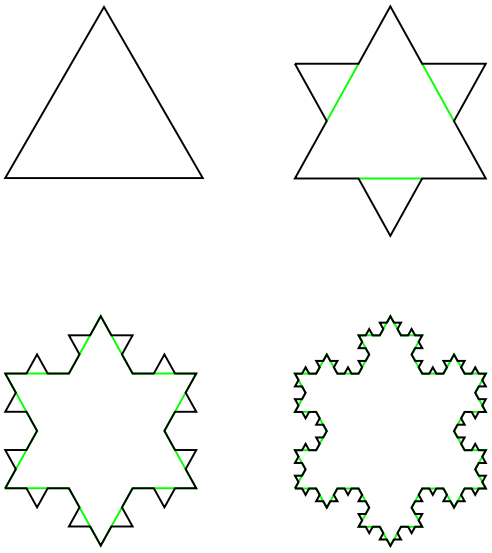

FIGURE 3 – Courbe de von Koch.

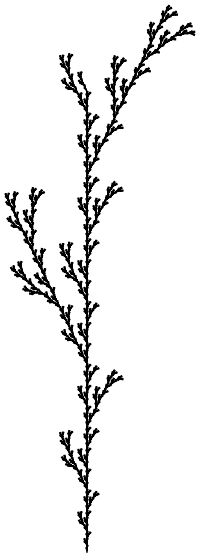

FIGURE 4 – Illustration tirée de [1].

III) Conversion en XML de L-systèmes au format CSV

Le fichier `l-systems.csv` fourni avec cet énoncé contient un ensemble de définitions de L-systèmes, un par ligne. La suite des valeurs de chaque ligne est de la forme suivante :

- le nom du système,
- la chaîne formée de ses symboles, sans répétitions ni espaces.
- son axiome,
- sa substitution, définie en spécifiant, pour chaque symbole, dans l'ordre d'apparition de ces symboles dans la chaîne précédente, l'image de ce symbole,
- son interprétation, définie en spécifiant, pour chaque symbole et dans un ordre similaire, l'instruction de la Tortue associée à ce symbole.

Noter que le nombre de valeurs sur chaque ligne est variable, puisque ces systèmes n'ont pas tous le même ensemble/nombre de symboles.

a) Étape préliminaire : conversion en XML des données brutes

Votre premier travail[2] sera de :

1. Choisir un format XML permettant de représenter un ensemble de L-systèmes, de symboles, axiomes, substitutions et interprétations quelconques.
2. Écrire dans le langage de votre choix (C, Java, OCaml, Python, ...) un programme prenant en entrée un fichier CSV au même format que celui fourni, et produisant un fichier XML au format que vous aurez choisi. Ce fichier devra contenir les descriptions de chacun des L-systèmes du CSV source.
3. Écrire un Schéma XML (fichier `xsd`) validant votre format.

2. Cette partie ne nécessite que de la programmation et les Schémas XML : elle peut être commencée dès la mise en ligne de ce sujet.

IV) Conversion des données XML en SVG

a) Étape 1 : extraction d'un script pour la Tortue

L'étape suivante sera d'écrire un fichier XSLT qui, à partir d'un fichier XML décrivant un ensemble de L-systèmes au format choisi précédemment, le nom de l'un ces systèmes (ou au moins, le rang du système dans le fichier) et un entier n, produira un nouveau fichier XML décrivant l'interprétation de l'itération de rang n du L-système choisi. Il vous faudra donc :

1. Choisir un format XML permettant de représenter un script pour la Tortue.
2. Écrire un Schéma XML validant le format que vous aurez choisi.

b) Étape 2 : conversion en un script pour le Traceur

L'avant dernière étape sera d'écrire un fichier XSLT qui, à partir d'un fichier XML décrivant un script pour la Tortue, produira un nouveau fichier XML décrivant un script équivalent pour le Traceur[3]. Là encore, il vous faudra :

1. Choisir un format XML permettant de représenter un script pour le Traceur.
2. Écrire un Schéma XML validant le format que vous aurez choisi.

c) Étape 3 : conversion au format SVG

Cette toute dernière étape sera la plus facile. Un script pour le Traceur n'est rien de plus, à la syntaxe près, que la description d'un chemin SVG (`<path d="..."/>`). Il s'agira d'écrire un dernier fichier XSLT qui, à partir d'un fichier XML décrivant un script pour le Traceur, produira un document SVG contenant un unique chemin similaire à celui tracé par le script.

Les dimensions souhaitées pour le document seront passées au processeur XSLT. Le chemin SVG devra être centré et occuper une surface maximale dans la page, mais sans déborder de la page. Pour effectuer cette conversion, il vous faudra sans doute définir une fonction XSLT récursive terminale.

V) Modalités

a) Binômes

Le projet doit être impérativement réalisé en binôme. Toutefois, en cas d'impossiblité majeure de satisfaire cette contrainte (nombre impair d'étudiants, conditions d'études particulières, etc), nous vous invitons à en discuter le plus tôt possible avec votre enseignant de cours magistral - aucun travail non réalisé en binôme ne sera accepté sans son accord explicite. La répartition des tâches au sein d'un binôme doit être raisonnablement équilibrée. En cas de déséquilibre avéré, les notes finales pourront être individualisées.

3. Vous aurez besoin, pour cette étape, de fonctions trigonométriques. Elle sont utilisables en ajoutant à la balise `<stylesheet>` du XSLT l'attribut `xmlns:math="http://www.w3.org/2005/xpath-functions/math"`. Ces fonctions peuvent ensuite s'écrire `math.cos(...)`, `math.sin(...)` (argument en `xsd:double` de résultat `xsd:double`), `math.pi()`, etc.

b) Usage de Gitlab.

Votre projet doit être réalisé à l'aide d'un dépôt sur le Gitlab de l'université [4]. Ce dépôt doit être privé, créé par l'un des deux membres d'un binôme, l'autre membre du binôme étant invité comme développeur. Vous donnerez accès (au moins comme reporter) à ce dépôt à chacun des enseignants du cours (et bien sûr à aucun autre utilisateur). La régularité et l'équilibrage des commits pourront être pris en compte dans l'évaluation.

Afin de ne pas alourdir inutilement votre dépôt, vous devrez mettre dans `.gitignore` tous les noms de fichiers dont le contenu pourra être généré par traitement : tous les `xml`, tous les `svg`.

c) Individualité de chaque projet

De manière évidente, votre code doit être strictement personnel. Tout partage de code entre binômes est évidemment interdit [5], et il relève de votre responsabilité de faire en sorte que votre code reste inaccessible aux autres binômes.

d) Forme du rendu

La date butoir pour votre dernier commit et celle des soutenances seront précisées ultérieurement. Votre rendu consistera en les éléments suivants, placés sur la branche `master` de votre dépôt :

- Les fichiers demandés (le code de votre extracteur, les fichiers `xsd` et `xsl`, mais aucun `xml` ni `svg`, voir ci-dessus), ainsi que tout autre fichier utile (*e.g.* un `Makefile`).
- Un fichier texte nommé `README` contenant vos noms et numéros d'étudiants, et précisant la manière d'utiliser ces fichiers.
- Un rapport de quelques pages au format PDF décrivant votre projet, et expliquant et justifiant les choix de conception ou d'implémentation, la répartition et la chronologie des tâches, les difficutés rencontrées, les parties non réalisées, les bugs non résolus, etc.

Références

[1] Przemysław Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants.* Springer-Verlag. 2004. http://algorithmicbotany.org/papers/#abop

4. https://gaufre.informatique.univ-paris-diderot.fr/users/sign_in

5. La soutenance de projet est un examen comme un autre. Le plagiat en projet constitue une fraude aux examens passible de lourdes sanctions discplinaires – ce cas s'est produit hélas plus d'une fois dans cette UFR. Par ailleurs, compte tenu des libertés de choix laissées par l'énoncé, il est assez improbable que deux rendus distincts soient accidentellement très similaires.