XML : Rapport de projet

Sriguru ELUMALAI, Jean-Philipp MAMBILA TETE

May 19, 2024

1 Introduction

Les L-Systèmes, ou systèmes de Lindenmayer, sont des modèles mathématiques initialement conçus pour simuler la croissance des plantes.

Ce projet vise à explorer et à implémenter les L-Systèmes à travers plusieurs étapes clés : la conversion de données brutes au format CSV en XML, l'extraction et la transformation de scripts pour la Tortue et le Traceur, et enfin la génération de graphiques au format SVG. Le but est donc de créer un outil flexible et efficace permettant de visualiser des structures complexes dérivées des L-Systèmes.

Ce rapport documente en détail les méthodes et les technologies utilisées pour atteindre ces objectifs, les défis rencontrés durant le développement, ainsi que les solutions apportées. Il fournit également une vue d'ensemble des résultats obtenus.

2 Implémentation

2.1 Conversion en XML de L-Systèmes au format CSV

La première étape de notre projet consiste à convertir les données des L-Systèmes, fournies sous forme de fichiers CSV, en un format XML structuré. Ce processus implique la lecture du fichier CSV, l'extraction des données pertinentes, puis la création d'un fichier XML avec une structure spécifique.

Voici le fichier CSV contenant les définitions des L-Systèmes :

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11	Column12	Column13
snow	APM	APFAPFA	AMAPFAMA	P	M	LINE 30	TURN 60	TURN -60				
koch	APM	APAPAPA	AMAAPAAPAPAMAMAAPAPAMAMAAMAAPA	P	M	LINE 10	TURN 90	TURN -90				
koch1	ABPM	APAPAPA	APBMAAPAPAAPABPAAMBPAAMAMAAAMBMAAA	BBBBBB	P	M	LINE 10	MOVE 10	TURN 90	TURN -90		
koch2	APM	APAPAPA	AAPAPAPAPAPAMA	P	M	LINE 10	TURN 90	TURN -90				
koch3	AP	APAPAPA	AAPAPAPAPAA	P	LINE 10	TURN 90						
koch4	APM	APAPAPA	AAPAMAPAPAA	P	M	LINE 10	TURN 90	TURN -90				
koch5	APM	APAPAPA	AAPAPPAPA	P	M	LINE 10	TURN 90	TURN -90				
dragon	ABPM	A	APBP	MAMB	P	M	LINE 10	LINE 10	TURN 90	TURN -90		
sierp	ABPM	B	BPAPB	AMBMA	P	M	LINE 6	LINE 6	TURN 60	TURN -60		
gosp6	ABPM	A	APBPPBMAMMAAMBP	MAPBBPPBPAMMAMB	P	M	LINE 4	LINE 4	TURN 60	TURN -60		
gosp4	ABPM	MB	AAMBMBPAPAMBMBAPBPAABMAPBPAAPBMABMBMAPAPBBM	PAAMBMBPAPABPAMBBMAMBPABBMAMBAPAPBMBMAPAPBB	P	M	LINE 2	LINE 2	TURN 90	TURN -90		
br1	APM	A	A[P[A]A[MA]A	P	M	LINE 5	TURN 25	TURN -25				
br2	APM	A	A[P[A]A[MA]A[A]	P	M	LINE 5	TURN 20	TURN -20				
br3	ABPM	A	B[P[A]B[MA]BA	BB	P	M	LINE 5	LINE 5	TURN 25	TURN -25		
htree	ABCPM	A	B[P[A]MA	CC	B	P	M	LINE 10	LINE 14	LINE 10	TURN 90	TURN -90

Le programme en C (`csv_to_xml.c`) réalise la conversion du CSV en XML en trois étapes clés :

- **Lecture du Fichier CSV** : Le fichier CSV est lu en mémoire, et chaque ligne est analysée pour en extraire les valeurs séparées par des virgules.
- **Stockage des Données** : Les valeurs extraites sont stockées dans une structure de données (`csvString`) qui contient les valeurs et leurs séparations.
- **Génération du Fichier XML** : Le programme parcourt les données stockées et génère un fichier XML en suivant une structure prédéfinie.

Le format XML que nous avons choisi pour représenter les L-Systèmes est la suivante :

```xml
<?xml version="1.0" encoding="iso-8859-1"?>
<LSystems>
    <!-- Premier L-système -->
    <LSys id="LSys1">
        <Name>snow</Name>
        <Symbols id="Symbols1">
            <Symbol id="Symbol1">A</Symbol>
            <Symbol id="Symbol2">P</Symbol>
            <Symbol id="Symbol3">M</Symbol>
        </Symbols>
        <Axiom>APPAPPA</Axiom>
        <Substitutions id="Subst1">
            <Substitution>
                <SymbolRef idref="Symbol1"/>
                <Image>AMAPPAMA</Image>
            </Substitution>
            <Substitution>
                <SymbolRef idref="Symbol2"/>
                <Image>P</Image>
            </Substitution>
            <Substitution>
                <SymbolRef idref="Symbol3"/>
                <Image>M</Image>
            </Substitution>
        </Substitutions>
        <Interpretations id="Interp1">
            <Interpretation>
                <SymbolRef idref="Symbol1"/>
                <Instruction>LINE 30</Instruction>
            </Interpretation>
            <Interpretation>
                <SymbolRef idref="Symbol2"/>
                <Instruction>TURN 60</Instruction>
            </Interpretation>
            <Interpretation>
                <SymbolRef idref="Symbol3"/>
                <Instruction>TURN -60</Instruction>
            </Interpretation>
        </Interpretations>
    </LSys>
</LSystems>
```

Explication de la structure XML ci-dessus :

- **La racine <LSystems>** : La balise racine <LSystems>encapsule tous les L-Systèmes contenus dans le document.
- **L-Système Individuel <LSys>** : Chaque L-Système est encapsulé dans une balise <LSys>, identifiée par un attribut id unique.
- **Nom du L-Système <Name>** : Représente le nom de chaque L-Système.
- **Définition des Symboles <Symbols>** : Les symboles utilisés dans le L-Système sont listés dans une balise <Symbols>, chaque symbole étant encapsulé dans une balise <Symbol>avec un attribut id unique.
- **Axiome <Axiom>** : L'axiome, ou la chaîne de départ du L-Système, est défini dans une balise <Axiom>.
- **Substitutions <Substitutions>** : Les règles de substitution sont définies dans une balise <Substitutions>. Chaque règle de substitution est encapsulée dans une balise <Substitution>et contient une référence au symbole <SymbolRef>et l'image de remplacement <Image>.

- **Interprétations <Interpretations>** : Les instructions d'interprétation pour les symboles sont définies dans une balise <Interpretations>. Chaque instruction est encapsulée dans une balise <Interpretation>et contient une référence au symbole <SymbolRef>et une instruction <Instruction>.

2.2 Extraction d'un script pour la Tortue

Pour extraire un script pour la tortue à partir d'un L-Système, nous avons utilisé une feuille de transformation XSLT. Cette feuille de transformation prend un document XML décrivant un L-Système et génère un script d'instructions pour la Tortue.

Le fichier XSLT (turtle-script.xsl) prend en paramètre le nom du L-Système à transformer (lysName) et un entier (n) pour spécifier le nombre d'itérations à appliquer. Par default il prend le 1er L-Système dans le document XML de départ et applique 2 itérations :

```
1 <xsl:param name="lsysName" as="xsd:string" select="//LSys[1]/Name"/>
2 <xsl:param name="n" as="xsd:integer" select="2"/>
```

Les fonctionnalités de ce XSLT sont les suivantes :

- **applyIterations** : Cette fonction applique récursivement les itérations sur l'axiome du L-Système en utilisant les substitutions définies.

```
1 <xsl:function name="my:applyIterations" as="xsd:string">
2 <!-- Contenu de la fonction -->
3 </xsl:function>
```

- **applySubstitutions** : Cette fonction applique les substitutions aux symboles de l'axiome pour générer le nouvel axiome après chaque itération.

```
1 <xsl:function name="my:applySubstitutions" as="xsd:string">
2 <!-- Contenu de la fonction -->
3 </xsl:function>
```

- **interpretSymbols** : Cette fonction interprète les symboles résultants pour générer les instructions correspondantes pour la tortue.

```
1 <xsl:function name="my:interpretSymbols" as="element(Instruction)*">
2 <!-- Contenu de la fonction -->
3 </xsl:function>
```

- **Template principal** : Le template principal sélectionne le L-Système correspondant, récupère l'axiome initial, applique les itérations et génère les instructions pour la tortue.

```
1 <xsl:template match="/">
2 <!-- Contenu du modèle principal -->
3 </xsl:template>
```

Le format XML que nous avons choisi pour représenter le script pour la Tortue est la suivante :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TortueScript xmlns:my="http://example.com/myfunctions"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <Instructions>
      <Instruction>
         <Type>LINE</Type>
         <Value>30</Value>
      </Instruction>
      <Instruction>
         <Type>TURN</Type>
         <Value>60</Value>
      </Instruction>
      <Instruction>
         <Type>TURN</Type>
         <Value>60</Value>
      </Instruction>
      <Instruction>
         <Type>LINE</Type>
         <Value>30</Value>
      </Instruction>
      <Instruction>
         <Type>TURN</Type>
         <Value>60</Value>
      </Instruction>
      <Instruction>
         <Type>TURN</Type>
         <Value>60</Value>
      </Instruction>
      <Instruction>
         <Type>LINE</Type>
         <Value>30</Value>
      </Instruction>
   </Instructions>
</TortueScript>
```

Explication de la structure XML ci-dessus :

- **La racine <TortueScript>** : Élément racine du document XML. Il contient toutes les instructions que la tortue doit suivre.
- **La balise <Instructions>** : Conteneur pour toutes les instructions. Chaque instruction individuelle est un sous-élément de cet élément.
- **La balise <Instruction>** : Élément qui décrit une instruction unique pour la tortue. Il contient deux sous-éléments : Le type <Type>d'instruction que la tortue doit exécuter (par exemple, LINE, TURN, STORE, RESTORE) et la valeur <Value>associée à l'instruction. Cela peut être un angle de rotation ou une longueur de ligne, selon le type d'instruction.

2.3 Conversion en un script pour le Traceur

Pour convertir les instructions générées par notre script pour la tortue en un script pour le Traceur, nous avons défini un format XML spécifique et une feuille de transformation XSLT.

Le fichier XSLT (tracer-script.xsl) permet de prendre les instructions de mouvement et de rotation de la Tortue et de les traduire en commandes de dessin pour le Traceur.

Les fonctionnalités de ce XSLT sont les suivantes :

- **Template principal** : Ce template principal initialise les variables pour les coordonnées et l'angle de la tortue, puis appelle le premier élément <Instruction>pour commencer le traitement.

```xml
<xsl:template match="/">
<!-- Contenu du modèle principal -->
</xsl:template>
```

- **Template de traitement des instructions** : Ce template traite chaque instruction individuellement, en fonction des actions que la tortue peut effectuer telles que le déplacement, la rotation, le stockage et la restauration.

```xml
<xsl:template match="Instruction">
    <!-- Contenu du modèle de traitement des instructions -->
</xsl:template>
```

- **Template pour le traitement des instructions de type LINE** : Ce template calcule les nouvelles coordonnées et crée un élément XML correspondant à une ligne.

```xml
<xsl:template name="ProcessLine">
    <!-- Contenu du modèle pour le traitement des instructions de type
LINE -->
</xsl:template>
```

- **Template pour le traitement des instructions de type MOVE** : Similaire au précédent, ce template calcule les nouvelles coordonnées et crée un élément XML correspondant à un déplacement.

```xml
<xsl:template name="ProcessMove">
    <!-- Contenu du modèle pour le traitement des instructions de type
MOVE -->
</xsl:template>
```

- **Template pour le traitement des instructions de type TURN** : Ce template met à jour l'angle en ajoutant la valeur de rotation spécifiée dans l'instruction.

```xml
<xsl:template name="ProcessTurn">
    <!-- Contenu du modèle pour le traitement des instructions de type
TURN -->
</xsl:template>
```

Le format XML que nous avons choisi pour représenter le script pour le traceur est la suivante :

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TraceurScript xmlns:math="http://www.w3.org/2005/xpath-
functions/math">
    <LINETO x="30" y="0"/>
    <LINETO x="45" y="-25.98076211"/>
    <LINETO x="60" y="0"/>
    <LINETO x="90" y="0"/>
    <LINETO x="75" y="25.98076211"/>
    <LINETO x="90" y="51.96152422"/>
    <LINETO x="60" y="51.96152422"/>
    <LINETO x="45" y="77.94228633"/>
    <LINETO x="30" y="51.96152422"/>
    <LINETO x="0" y="51.96152422"/>
    <LINETO x="15" y="25.98076211"/>
    <LINETO x="0" y="0"/>
</TraceurScript>
```

Explication de la structure XML ci-dessus :

- **La racine <TraceurScript>** : Élément racine du document XML pour le traceur. Contient toutes les commandes de dessin.
- **La balise <LINETO>** : Instruction pour dessiner une ligne jusqu'aux coordonnées spécifiées (x, y).
- **La balise <MOVETO>** : Instruction pour déplacer la position actuelle sans dessiner une ligne jusqu'aux coordonnées spécifiées (x, y).
- **Les balises <STORE>et <RESTORE>** : Instructions pour sauvegarder et restaurer l'état actuel (position et angle). Ces balises sont présentes mais n'ont pas d'attributs.

2.4 Conversion au format SVG

Pour générer un dessin au format SVG à partir du script Traceur, nous avons développé une feuille de transformation XSLT nommée `convertToSVG.xsl`. Cette feuille de style prend en charge la création d'un SVG à partir des instructions de dessin fournies par le script Traceur.
Les fonctionnalités de ce XSLT sont les suivantes :

- **Template racine du script Traceur** : Calcule les coordonnées minimales et maximales pour déterminer la hauteur et la largeur du dessin SVG, puis génère le chemin SVG en appliquant les templates MOVETO et LINETO.

```
1 <xsl:template match="/TraceurScript">
2   <!-- Contenu du modèle principale -->
3 </xsl:template>
```

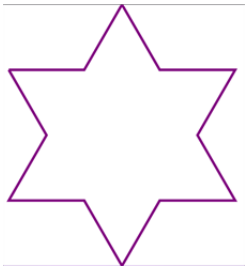
- **Template pour MOVETO** : Ajuste les coordonnées en fonction des coordonnées minimales calculées précédemment et les ajoute au chemin SVG.

```
1 <xsl:template match="MOVETO">
2   <!-- Contenu du modèle MOVETO -->
3 </xsl:template>
```

- **Template pour LINETO** : Ajuste les coordonnées et les ajoute au chemin SVG.

```
1 <xsl:template match="LINETO">
2   <!-- Contenu du modèle LINETO -->
3 </xsl:template>
```

Résultat de notre exemple sur le 1er L-Système à 2 itérations :



3 Calendrier Jalons

La répartition des tâches entre les membres du binôme s'est déroulée de manière suivante :

Répartitions tâches	Développement du programme CSV en XML	Fichier XSLT pour la Tortue	Conversion du script pour le Traceur	Conversion au format SVG
Jean-Philipp	100%	0%	100%	0%
Sriguru	0%	100%	0%	100%

Le projet a suivi un calendrier bien défini, avec plusieurs jalons atteints à chaque étape :

4 Défis rencontrés

Durant le développement de ce projet, nous avons rencontré plusieurs défis significatifs :

- **Gestion des itérations** : La gestion des itérations dans les L-Systèmes, en particulier lors de la transformation en XML et lors de l'extraction du script pour la tortue, a été complexe en raison de la récursivité et de la nécessité de maintenir les données cohérentes à chaque étape.
- **Gestion du centrage du chemin du path** : Nous avons rencontré des défis pour centrer correctement le chemin du path dans le fichier SVG généré. Le but était d'éviter les débordements et de garantir un positionnement central du dessin. Nous avons finalement résolu ce problème en utilisant la vue (viewbox) pour ajuster correctement la position du chemin du path.
- **Gestion des instructions STORE et RESTORE** : Nous avons rencontré des difficultés dans la gestion des instructions STORE et RESTORE pour les L-Systèmes qui les incluent. En conséquence, les L-Systèmes br1, br2, br3 et htree afficheront uniquement leur branche principale, sans prendre en compte ces instructions. Cela limite la complexité des structures affichées pour ces L-Systèmes.
- **Optimisation des performances** : Malgré nos efforts, nous avons rencontré des difficultés à optimiser les performances de notre application, notamment lors de la génération de graphiques au format SVG pour des L-Systèmes complexes et un grand nombre d'itérations.

5 Conclusion

Ce projet nous a permis d'explorer en profondeur les L-Systèmes et les techniques de manipulation de données XML. En surmontant les défis rencontrés et en développant des solutions efficaces, nous avons réussi à créer un outil robuste pour la visualisation de structures complexes dérivées des L-Systèmes. L'utilisation de feuilles de transformation XSLT s'est avérée particulièrement puissante pour la conversion entre différents formats de données. En fin de compte, nous sommes satisfaits des résultats obtenus et de notre capacité à relever les défis techniques rencontrés tout au long du projet.

6 Annexes

Voici les résultats de chaque L-Système avec un nombre élevé d'itérations (sans STORE et RESTORE) :

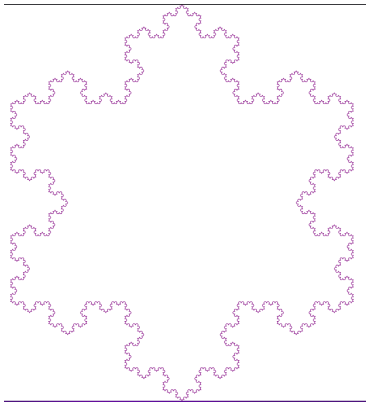


Figure 1: snow avec 6 itérations

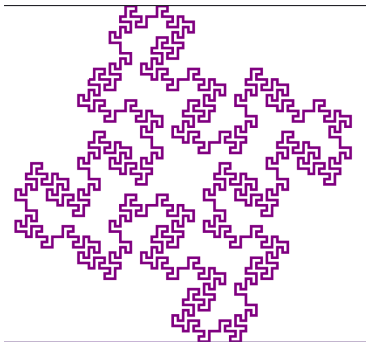


Figure 2: koch avec 3 itérations

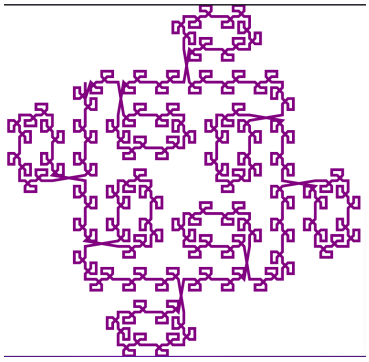


Figure 3: koch1 avec 3 itérations

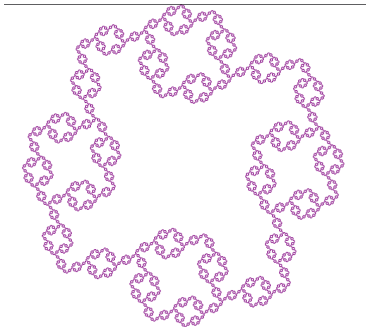


Figure 4: koch2 avec 6 itérations

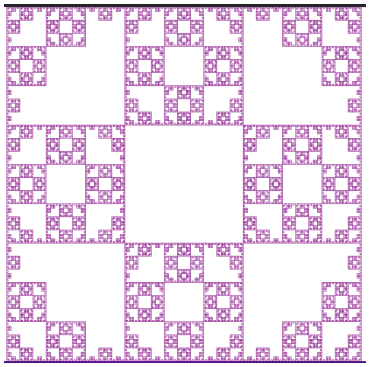


Figure 5: koch3 avec 6 itérations

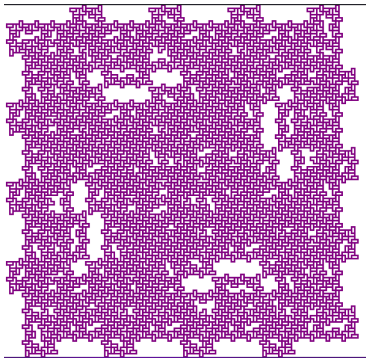


Figure 6: koch4 avec 6 itérations

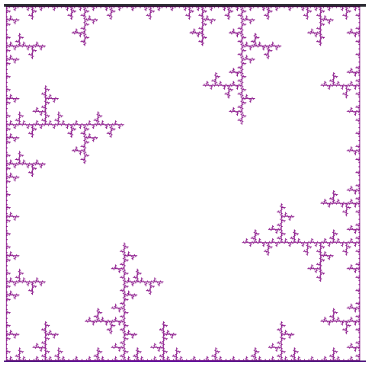


Figure 7: koch5 avec 6 itérations

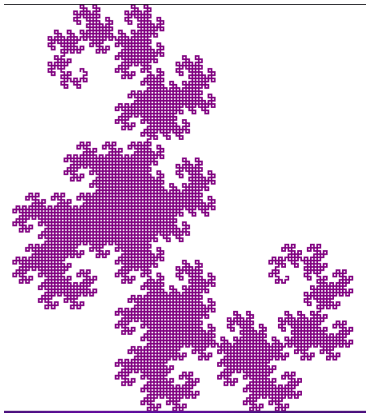


Figure 8: dragon avec 14 itérations

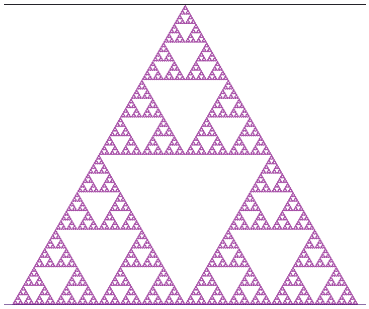


Figure 9: sierp avec 10 itérations

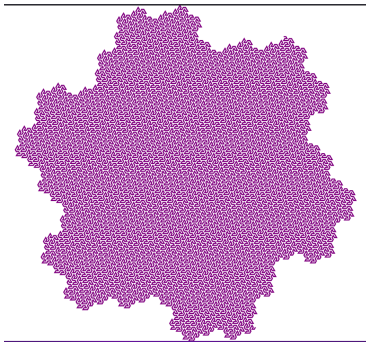


Figure 10: gosp6 avec 6 itérations

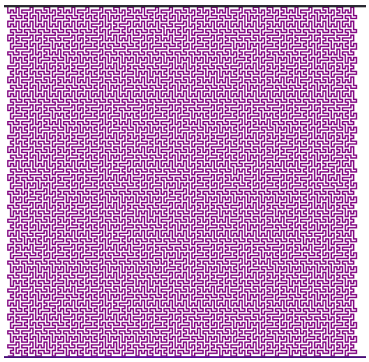


Figure 11: gosp4 avec 4 itérations

Résultats des L-Systèmes avec la gestion des STORE et RESTORE non résolu :

Figure 12: br1 avec 5 itérations

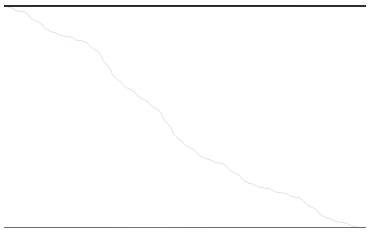


Figure 13: br2 avec 5 itérations

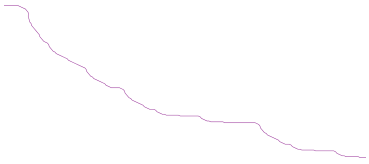


Figure 14: br3 avec 5 itérations

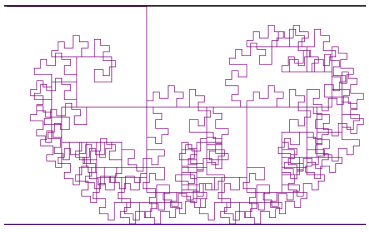


Figure 15: htree avec 10 itérations