

LICENCE 2 INFORMATIQUE

Générateur de flores ludiques L-Système

Auteur :

DIALLO ELHADJ ALSEINY
DJIGUINE MAMADY
DIALLO ABDOULAYE DJIBRIL
DIALLO MAMADOU ALPHA

Chargé du cours :

Bonnet GRÉGORY

Encadrant TP :

Dauprat QUENTIN

5 avril 2022

Table des matières

1	Introduction	2
1.1	Sujet et Grandes Lignes	2
1.2	Mise en Place du projet	3
2	L-Système	4
2.1	Définition et Fonctionnement	4
2.1.1	De quoi s'agit-il ?	4
2.1.2	Principe de Fonctionnement	4
2.2	L-Système Utilisé	4
3	Organisation du Projet	6
3.1	Répartition des tâches	6
3.2	Architecture du Programme	6
4	Élément Technique	8
4.1	Parseur	8
4.2	Rule	9
4.3	Interface	9
4.3.1	FenetrePrincipale :	9
4.3.2	Configuration :	9
4.3.3	Menu :	9
4.3.4	Rendu :	9
4.4	Moteur graphique	9
4.4.1	Rendu2D	9
4.4.2	Rendu3D	10
5	Expérimentations et Usages	11
5.1	Utilisation	11
5.2	Lancement de l'application	11
5.3	Fonctionnement de l'interface Graphique	11
5.4	Navigation dans l'interface graphique 3D	12
5.5	Test du logiciel	13
5.5.1	Possible problème	13
5.5.2	Test du logiciel	13
6	Conclusion	14
6.1	Bilan de nos travaux	14
6.2	Améliorations possibles	14

Chapitre 1

Introduction

1.1 Sujet et Grandes Lignes

Dans toute entité de quelque nature que ce soit ,le travail en équipe s'avère indispensable entre les personnes qui la composent afin de parvenir à un meilleur résultat , c'est dans ce cadre qu'il nous a été proposé plusieurs sujets afin de travailler en un groupe de quatre (4) étudiants sur un de ces sujets et réaliser une application bout en bout dans un environnement orienté objet (java) en appliquant les notions de la programmation orientée objet et ses principes vues durant tout le semestre et plus particulièrement en conception de logicielle. Parmi Les sujets proposés celui qui a le plus retenu notre attention est le Générateur de flores vidéos-ludiques dont l'énoncé est ci-dessous : « Il n'est pas rare de rencontrer des arbres, buissons ou plantes, plus ou moins réelles, dans des jeux vidéos ou des films d'animation. Les L-systèmes permettent de représenter ces modèles végétaux sous formes de système de réécriture. Le but de ce projet est donc de réaliser un simulateur de L-système végétal qui prend des règles de réécritures en entrée et produit une image 2D (ou une scène 3D dans un second temps) de l'objet obtenu par la simulation de ce système. Il faudra donc implémenter un parseur de L-système, un moteur de réécriture, puis un moteur de rendu graphique pour visualiser ces plantes. » Les grandes lignes de ce sujet étant :

- Implémentation d'un parseur de l-système permettant de représenter tout lsysteme végétal ;
- Faire un rendu 2D en fonction du résultat de réécriture issue de la génération d'un l-système ;
- Et en fin produire la même représentation en scène de 3D .

1.2 Mise en Place du projet

Pour la mise en place du projet ,nous nous sommes dans un premier orienté dans une optique de compréhension générale des L-systèmes particulièrement un l-système végétal grâce aux liens proposés en dessous du sujet dont le second proposant un livre "**The algorithmic beauty of plants**" disponible en ligne décrivant tous les l-systèmes afin de pouvoir adopter un modèle nous permettant de mettre en place notre parseur et notre modèle de réécriture. Ensuite nous nous sommes renseignés sur les différents moteurs de rendu graphique 2D et 3D dont notre choix s'est finalement porté sur java graphique 2D pour le rendu 2D et JOGL¹ pour le rendu 3D proposé en dessous du sujet.

Après avoir compris le vif du sujet ,nous avons décidé de démarrer le projet selon un ordre de priorité nous permettant d'avancer de façon à mieux gérer notre temps ,en commençant par le parseur et son modèle de réécriture , le moteur de rendu 2D et l'interface graphique avec une option aide décrivant notre l- système pour un usage facile.

En fin nous nous sommes tous focalisé sur la dernièrement partie qui concerne le rendu 3D .

Notre objectif étant de réaliser un rendu 2D et 3D comme illustrent les deux images suivantes :

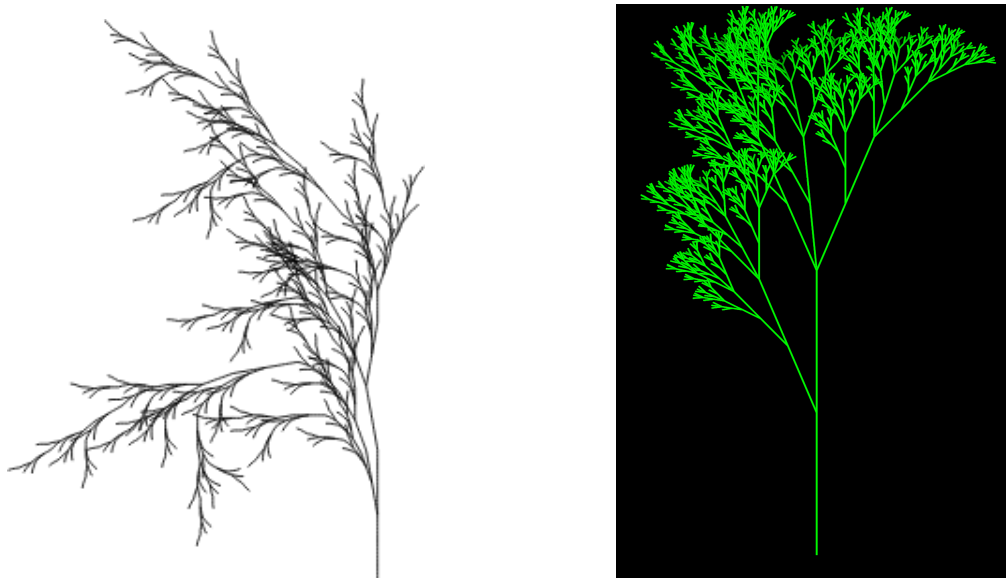


FIGURE 1.1 – Arbre en 2D et Arbre en 3D

1. (Java Open Graphics Library)

Chapitre 2

L-Système

2.1 Définition et Fonctionnement

2.1.1 De quoi s'agit-il ?

Un L-System (ou système de Lindenmayer) inventé en 1968 par le biologiste hongrois Aristid Lindenmayer¹ est un ensemble de règles et de symboles qui modélisent un processus de croissance d'êtres vivants comme des plantes ou des cellules.

2.1.2 Principe de Fonctionnement

Un L-système est un système de réécriture qui comprend :

- **Un alphabet** : l'ensemble des variables et constantes du L-Système ;
- **Un axiome** : qui représente le point de départ c'est à dire l'état initial.
- **Un ensemble de règles** : permettant à chaque étape de substituer les variables par leurs correspondants. Ainsi ,on obtient un L-système part toujours d'un axiome de départ avec un ensemble de règle qui vont être appliquée à chaque itération (entier de valeur donnée) jusqu'à trouver le mot résultat par substitution et qui sera ensuite interprété pour être interprété sous forme de rendu graphique 2D ou 3D

2.2 L-Système Utilisé

Un l-système étant essentiellement composé de lettres et symboles, notre l- système végétal utilise les lettres de l'alphabet et des symboles ci-dessous :

1. Lindenmayer Biologiste hongrois

Symboles	Interprétations
$\{\text{LETTRE}\} \setminus \{H, B\}$	Avancer d'une unité (dessine une branche)
+	Tourner à gauche d'un angle
-	Tourner à droite d'un angle
	Enregistrer la position et l'orientation
]	Rétablir la position et l'orientation
>	Rouler à gauche d'un angle α
<	Rouler à droite d'un angle α
B	S'incliner vers le bas en 3D
H	S'incliner vers le haut en 3D

TABLE 2.1 – Interprétations des symboles

Nous avons proposé beaucoup d'exemples de L-system végétal afin d'illustrer plusieurs exemples dont entre autres nous avons :

Pour un nombre de trois d'itération appliqué a au système ci-haut le tableau ci-dessous détaille le résultat obtenue à chaque itération par application de la règle .

Axiome de Base :	X
Règle1 :	$X = F[+X][-X]FX$
Règle2 :	$F = FF$
iteration 0	$F[+X][-X]FX$
iteration1	$FF[+F[+X][-X]FX][-F[+X][-X]FX]FFF[+X][-X]FX$
iteration2	$FFFF[+FF[+F[+X][-X]FX][-F[+X][-X]FX]FFF[+X][-X]FX][-FF[+F[+X][-X]FX][-F[+X][-X]FX]FFF[+X][-X]FX]FFFFF[+F[+X][-X]FX][-F[+X][-X]FX]FFF[+X][-X]FX$

TABLE 2.2 – Exemple de génération

Chapitre 3

Organisation du Projet

3.1 Répartition des taches

Le projet étant essentiellement composé de trois grandes parties ,nous nous sommes dans un premier temps réunis afin de mieux comprendre le sujet et élaborer un plan de travail ,ce qui nous a permis de comprendre c'est quoi un l-système dans l'ensemble, de quoi a t on besoin pour le représenter mais aussi la notion de réécriture qui est fonction d'un axiome, d'un ensemble règles et un nombre d'itération puis en dernier nous avons aussi appris des notions d'interprétation de représentation du résultat de ce système de réécriture sous de rendu graphique 2D et 3D.

Mamady Djiguiné s'est chargé de développer le parseur et l'écriture de son test, pendant ce temps Abdoulaye Djibril travaillait sur la représentation en 2D grâce aux exemples de génération donné dans le livre "**The algorithmic beauty of plants**",de même Diallo Alseiny travaillait quand à lui sur l'interface graphique et le rendu 3D ; en fin Mamadou alpha Diallo s'est chargé de préparer les fichiers indispensables pour une bonne configuration de notre application (Ant ,manifeste) mais aussi du rapport de notre projet.

3.2 Architecture du Programme

Concernant l'architecture de notre projet, nous avons voulu la relier au sujet pour une compréhension facile.Cependant nous avons opté pour une décomposition en quatre (4) packages :

- **parser** : le package parser permet de générer un l-système sous forme de chaîné de caractères ;
- **graphique** : permet d'interpréter le résultat du parser en rendu 2D et 3D ;
- **utils** : permet de gérer les différents positionnement au niveau du rendu ;
- **interfaceuser** : gérant tout ce qui est interface graphique utilisateur.

Chapitre 4

Élément Technique

4.1 Parseur

le parseur comme son nom l'indique constitue le cœur de notre application grâce à sa méthode génération permet de produire une chaîne représentant un l-système végétal par réécriture en se servant d'un axiom de départ, un nombre d'itération et des règles créée à partir de la classe **Rule** qu'il contient sous forme de liste .

Pour qu'elle produise une chaîne représentant un l-système il faut se s'assurer que ses attributs aient été initialisés et qu'aucun d'entre eux ne soit nul pour ceux de type entier et non vide pour ceux de type chaîne. En plus de ses attributs sa méthodes génération se sert également d'une méthode **compare** pour substituée chaque caractère par la chaîne devant le substituer suivant les règles définies.

l'algorithme de la génération d'un l-système par réécriture

Algorithme 1 : Génération d'un l-système par réécriture

Entrées : axiom : *String* , rules : *List[Rule]* , nbIteration : *Integer*

Sortie : Chaîne complète après la dernière itération

```
1 pour  $i \leftarrow 0$  à nbIteration faire
2   production  $\leftarrow ""$ 
3   pour  $j \leftarrow 0$  à axiom.taille faire
4     hasBeenUpdated  $\leftarrow$  faux
5     pour Rule  $r$  : rules faire
6       si  $r.compare(axiom.charAt[j])$  alors
7         hasBeenUpdated  $\leftarrow$  vrai
8         production  $\leftarrow$  production +  $r.getItValues$ 
9         arreter
10      fin
11    fin
12    si !hasBeenUpdated alors
13      | production  $\leftarrow$  production + axiom.charAt[j]
14    fin
15  fin
16  axiom  $\leftarrow$  production
17 fin
18 retourner axiom
```

4.2 Rule

Elle permet de composer les règles d'un l-système que la classe parser utilisera pour produire un system sous forme de chaîne avant que cette dernière ne soit interprétée sous forme de rendu 2D ou 3D. Cette classe est très utile aussi car elle contient en son sein une méthode *compare* qui prends un caractère et retourne la chaîne devant le substituer lors d'une génération.

4.3 Interface

Nous avons utilisé l'interface java swing pour realiser l'interface utilisateur de notre application .Elle composée quatre (4) classes. une permettant d'afficher la fenetre principale, une autre permettant de faire les Menu ,une troisième permettant defaire tout ce qui est configuration .

4.3.1 FenetrePrincipale :

Rassemble tous les composants graphiques menu, configuration et rendu. Elle constitue l'élément centrale gérant le lien entre le parseur et les modes de rendu.

4.3.2 Configuration :

Définie les zones de paramétrage de l'application ainsi que les boutons de contrôle :

- **Générer** : permettant générer un rendu 2D ou 3D du l-système ;
- **newRule** : pour ajouter au tant de règles souhaitées ;
- **clear** : pour nettoyer les zone de textes et rendu .

4.3.3 Menu :

Contient une liste déroulante de quelques exemples de l-systèmes sous format 2D et 3D, ainsi que l'option aide qui explique le fonctionnement du système.

4.3.4 Rendu :

Contient deux cases à cocher **rendu2d**, **rendu3d** permettant de choisir un mode de rendu ainsi qu'une zone de rendu pour la visualisation.

4.4 Moteur graphique

En ce qui concerne nos moteurs de rendu, nous utilisons du java swing pour le rendu 2D et jogl (java open gl library) pour le rendu 3D .

4.4.1 Rendu2D

Nous utilisons du java swing pour faire le rendu 2D en redéfinissant la méthode *paintComponent(graphics 2d)* de celui-ci. g2d.drawLine pour tracer les lignes(branche), g2d.translate pour faire des translations, g2d.scale pour gérer la mise en gras et une liste pour sauvegarder les positions si nécessaire.

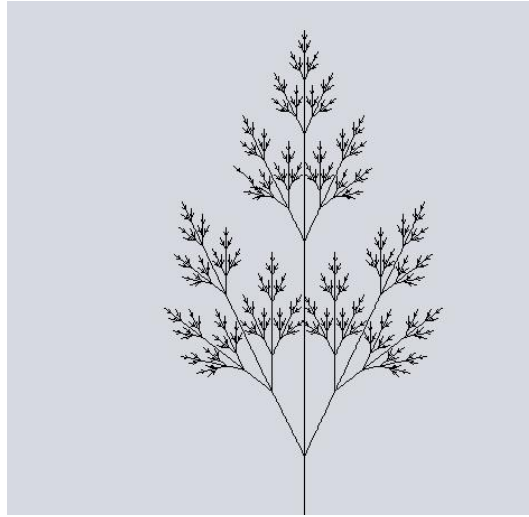


FIGURE 4.1 – Exemple de rendu 2D

4.4.2 Rendu3D

Comme mentionner ci-dessus, nous utilisons java openGL library pour faire notre rendu 3D notamment : gl2 pour afficher les lignes glu pour placer la caméra et GLUT pour afficher les cylindres. GLCanvas crée la fenêtre, GLEventListener initialise et affiche l'environnement 3D.

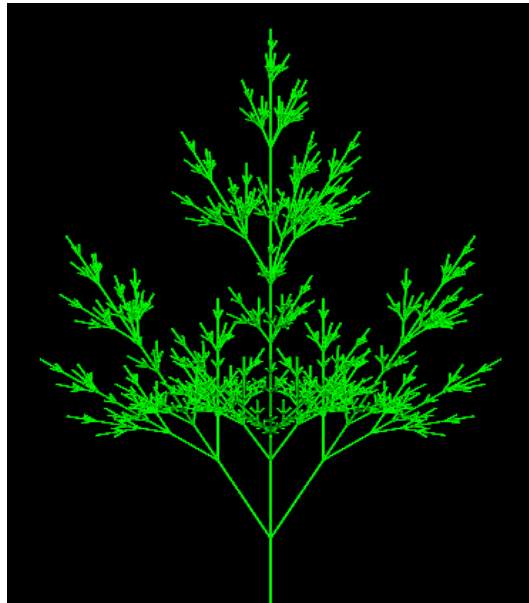


FIGURE 4.2 – Exemple de rendu 3D

Chapitre 5

Expérimentations et Usages

5.1 Utilisation

Notre avons développé notre application avec du java standard Édition 11 pour des raisons de compatibilité avec les ordinateurs de l'université. Elle peut s'exécuter sur les distributions Linux avec interface graphique tel que Ubuntu version 20.04.4 LTS, sur mac Os 12.3.1 et windows 10.

5.2 Lancement de l'application

Vous devez ouvrir un terminal à l'emplacement du dossier .

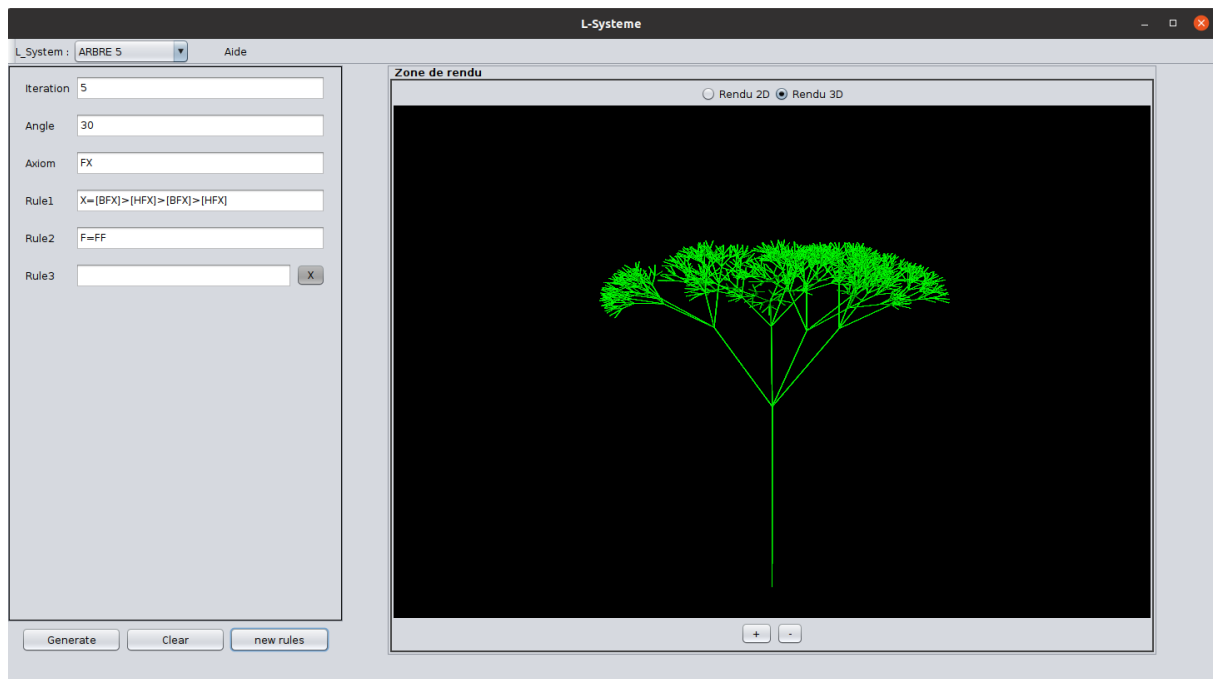
Pour :

- Lancer l'application exécuter : `ant run` ou `./scripts/run.sh`
- Initialiser le projet : `ant init` ou `fbox./scripts/install.sh`
Cette initialisation créera les dossiers de bases bin, doc et dist
- Compiler le projet : `ant compile` ou `./scripts/compile.sh`
- Générer le Javadoc : `ant javadoc` ou `./scripts/makedoc.sh`
- Générer le fichier jar : `ant packaging` ou `./scripts/makejar.sh`
- Exécuter le fichier jar : `./scripts/runjar.sh`
- Nettoyer le projet : `ant clean` ou `./scripts/clean.sh`
- Lancer le test : `ant test` ou `./scripts/test.sh`

5.3 Fonctionnement de l'interface Graphique

Au lancement du programme, il se présente avec un exemple par défaut visualisable par clic sur le bouton **Generate** sous forme 2D ou 3D.

Dans la barre de menu existe une liste déroulante contenant d'autres exemples ainsi qu'un bouton aide permettant à l'utilisateur de mieux comprendre le fonctionnement de l'interface et l'interprétation des symboles.



Il est possible qu'un utilisateur puisse composer ses propres règles. Pour cela les boutons **clear**, **Generate** et **Newrule** lui sera d'une aide (voir Configuration 4.3.2 page 9) pour l'utilisation des ces boutons.

Attention l'application tient compte de la caste. Toute modification entraînée sur l'interface graphique il faudrait cliquez sur le bouton **Generate** pour la visualiser.

Dans la zone rendu à gauche le rendu 2d est coché par défaut est vous pouvez le remettre sur le rendu 3D.

5.4 Navigation dans l'interface graphique 3D

Pour naviguer dans l'espace 3D les boutons : + ou - serviraient à zoomer ou dézoomer respectivement.

- Bouton + permet de zoomer
- Bouton - permet de dézoomer

En plus un clic glissé de la souris :

- Vers la droite : permet de faire tourner l'arbre autour de lui même dans la même direction ;
- Vers la gauche : permet de faire tourner l'arbre autour de lui même dans la même direction ;
- Vers le bas : incline l'arbre vers le bas

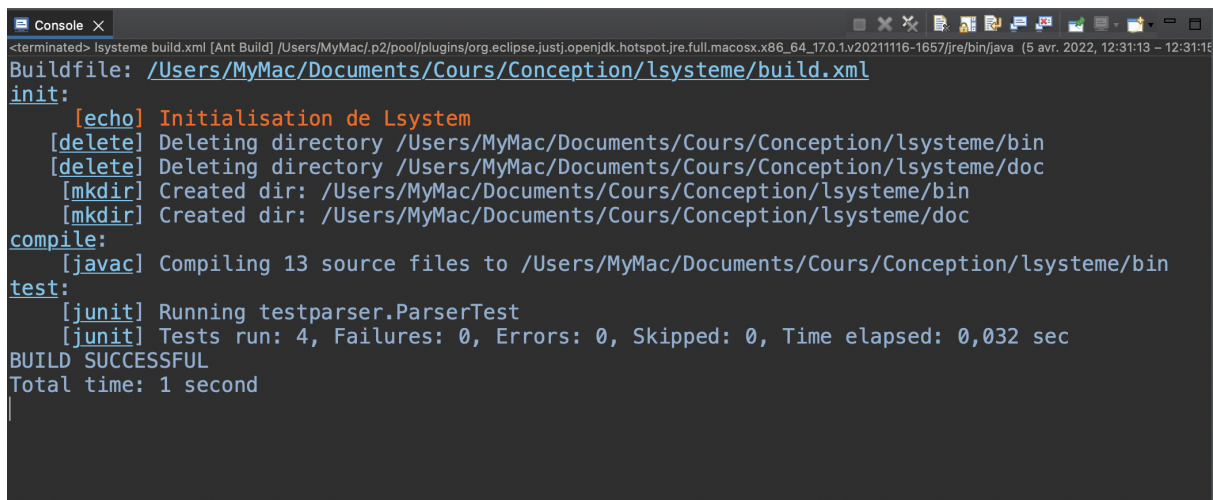
5.5 Test du logiciel

5.5.1 Possible problème

Lorsque vous lancer et essayer de générer un arbre avec un nombre d'itération au delà d'un certain seuil suivant la composition de l'arbre, cela pourrait prendre un peu de temps pour le parseur lors de la réécriture et sa représentation 3D voir 2D pourrait déborder. Dans ce cas une solution serait de diminuer le nombre d'itération, en plus en 3D on pourrait aussi dézoomer.

5.5.2 Test du logiciel

Nous avons réalisé nos tests en utilisant le framework open source **Junit-4.12** le développement et l'exécution de tests unitaires. Elle teste l'ensemble des méthodes indispensables pour la réécriture d'un arbre.



```
<terminated> lsysteme build.xml [Ant Build] /Users/MyMac/p2/pool/plugins/org.eclipse.justi.openjdk.hotspot.jre.full.macosx.x86_64_17.0.1.v20211116-1657/jre/bin/java (5 avr. 2022, 12:31:13 - 12:31:15)
Buildfile: /Users/MyMac/Documents/Cours/Conception/lsysteme/build.xml
init:
    [echo] Initialisation de Lsysteme
    [delete] Deleting directory /Users/MyMac/Documents/Cours/Conception/lsysteme/bin
    [delete] Deleting directory /Users/MyMac/Documents/Cours/Conception/lsysteme/doc
    [mkdir] Created dir: /Users/MyMac/Documents/Cours/Conception/lsysteme/bin
    [mkdir] Created dir: /Users/MyMac/Documents/Cours/Conception/lsysteme/doc
compile:
    [javac] Compiling 13 source files to /Users/MyMac/Documents/Cours/Conception/lsysteme/bin
test:
    [junit] Running testparser.ParserTest
    [junit] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0,032 sec
BUILD SUCCESSFUL
Total time: 1 second
```

FIGURE 5.1 – Test

Chapitre 6

Conclusion

6.1 Bilan de nos travaux

Étudier un sujet aussi passionnant que le l-système. Nous avons appris beaucoup de chose dans ce vaste domaine allant d'une expérience de modélisation aux différentes étape de création d'un arbre en passant par des exercices de programmation en Java. Il était très agréable de voir les idée théorique se concrétiser en rendu 2D et 3D. Nous retenons également de ce projet, la nécessité d'une bonne préparation, l'importance de la recherche et la gestion du temps pour la réalisation d'un programme.

6.2 Améliorations possibles

Nous pouvons continuer ce projet en de nombreuses façon :

- On aurait pu implémenter des l-système paramétrique et faire réagir l'arbre avec son environnement
- colorisation des arbre en 2D et 3D
- Possibilité d'ajouter plusieurs arbres dans la même fenêtre cote à cote
- Généraliser l'étude afin de pouvoir représenter n'importe quel l-système
- Implémenter des stochastiques l-système
- Ajouter des contrôles clavier sur le rendu 3D

Bibliographie

- [1] Wikipedia L-Système (FR) : <https://fr.wikipedia.org/wiki/L-Système> .
- [2] Wikipedia L-Système (EN) : <https://en.wikipedia.org/wiki/L-system> .
- [3] E. H. Norman <https://www.techno-science.net/definition/11374.html>
- [4] Bob Tadashi Wakabayashi <https://www.cs.unm.edu/~joel/PaperFoldingFractal/L-system-rules.html>
- [5] The algorithmic beauty of plants <http://algorithmicbotany.org/papers/abop/abop.pdf>
- [6] Moteur de rendu 3D <https://jogamp.org/jogl/www/>
- [7] JOGL Tutorial <https://www.javatpoint.com/jogl-tutorial>
- [8] Dessin java et java 2D <http://www-igm.univ-mlv.fr/~berstel/Cours/CoursJava/8-Dessin.pdf>
- [9] Java Platform Standard Edition 8 Documentation <https://docs.oracle.com/javase/8/docs/>
- [10] Stack overflow <https://stackoverflow.com/questions/18875244/java-build-ant-file-with-external-jar-files>
- [11] Houdini <https://www.sidefx.com/docs/houdini/nodes/sop/lssystem.html>
- [12] L-System Rules Parser <http://jeromeetienne.github.io/lssystem-js/parser-tool.html>
- [13] Vexlio <https://www.vexlio.com/blog/drawing-simple-organics-with-l-systems/>