Interpréteur de systèmes de Lindenmeyer

Dewanou Z. P. COTCHO & Boluwatife E. ADEKOYA &
Aissatou DIALLO & Jeremie LAYIZIA

Université de Caen Normandie

16 avril 2024

Table des matières

Introduction

Introduction

- Les systèmes de Lindenmeyer, ou L-systèmes, permettent de représenter des modèles de végétaux sous forme de système de réécriture.
- C'est un modèle formel utilisé pour décrire et simuler la croissance de structures biologiques telles que les plantes, les algues et autres.

Introduction

Objectifs

- Définir un langage pour construire un système de Lindenmeyer classique ;
- Implémenter un interpréteur pour une visualisation 2D et 3D ;
- Etendre le langage et les interpréteurs pour des systèmes stochastiques et/ou contextuels.

Organisation du code

Packages

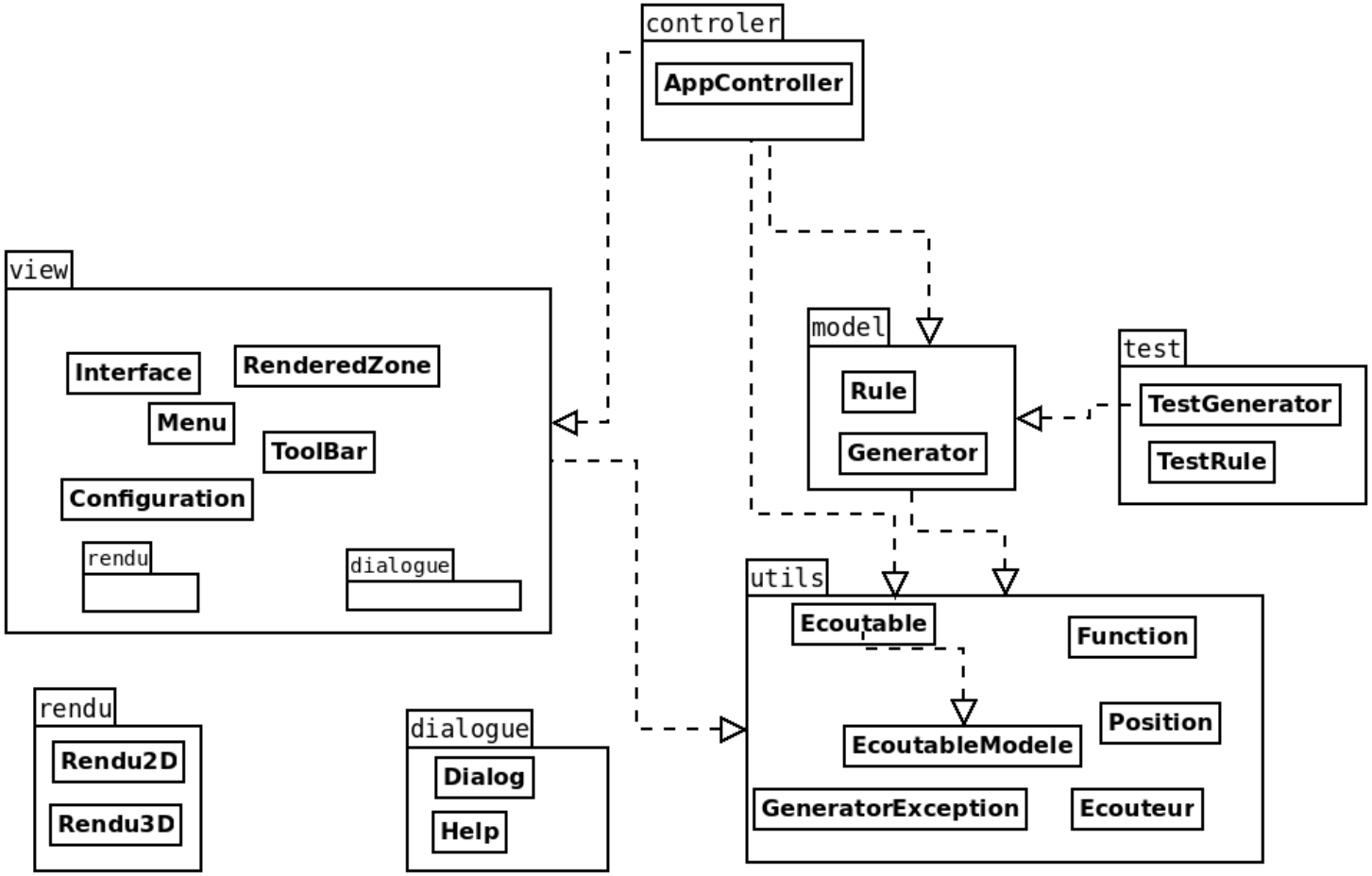

Figure – Packages

Retour sur le package model

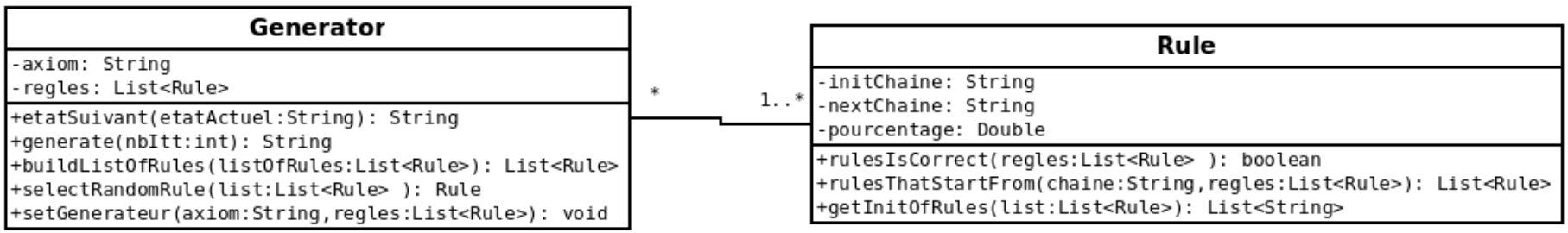

Figure – Les classes du package model

Retour sur le package view

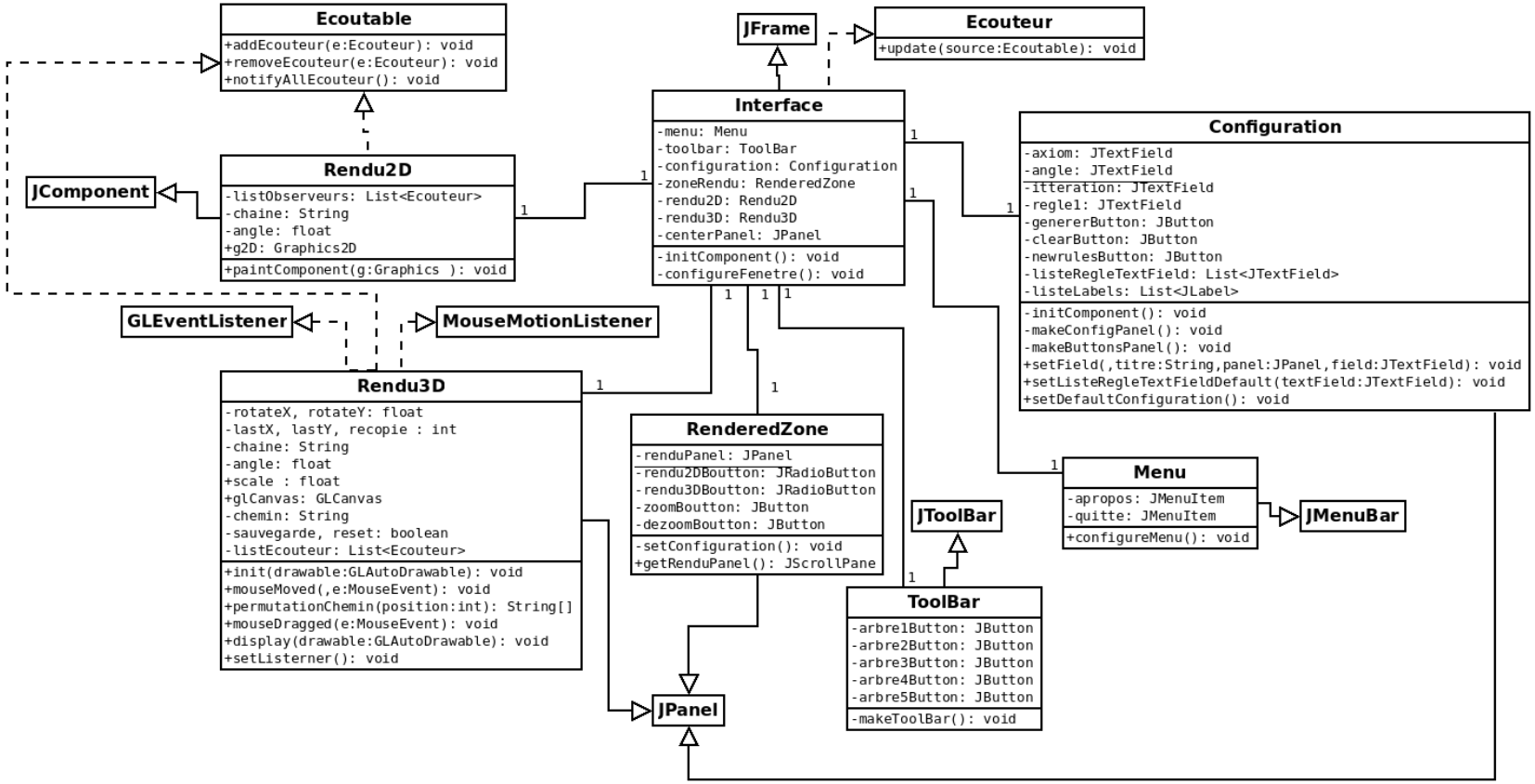

Figure – Les classes du package view

Retour sur le package controler

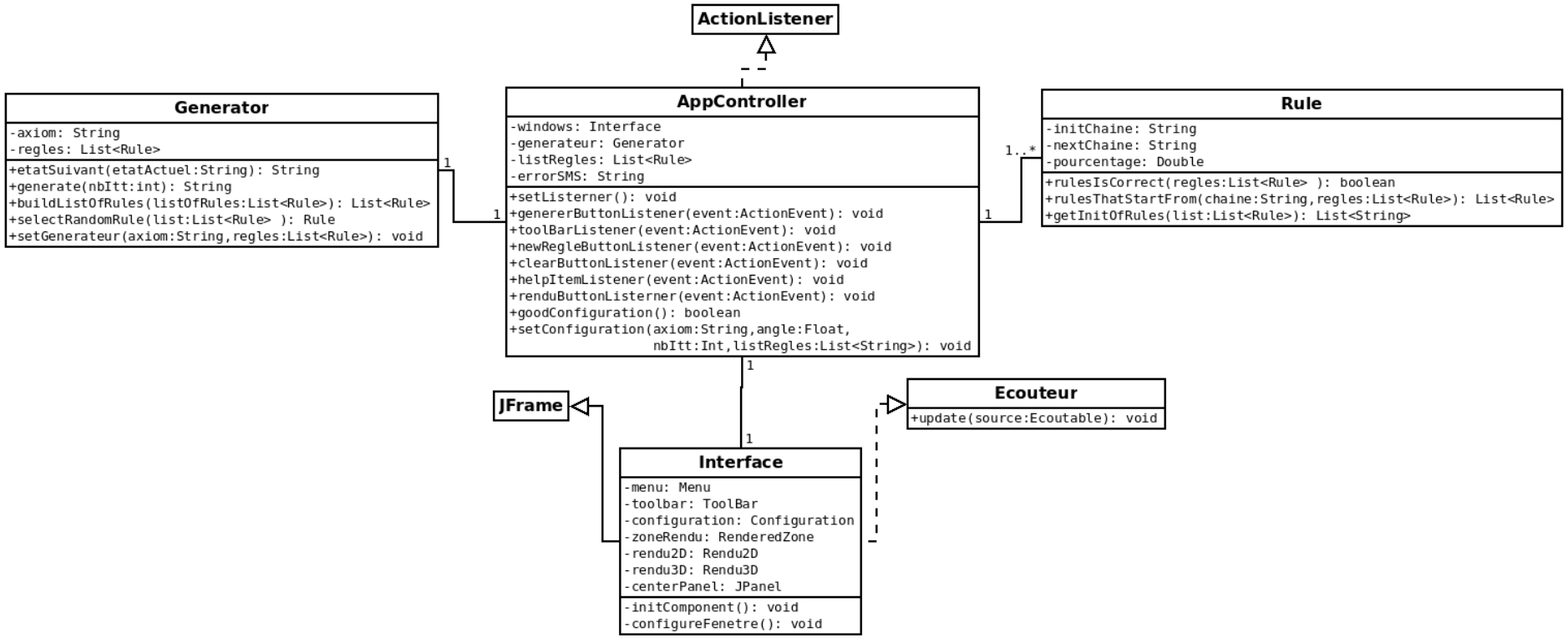

Figure – Package controler et interactions avec les autres classes

Explication de quelques méthodes

Calcul de la chaine interprétée en 2D et 3D

Algorithme 1 : Méthode **generate** : calcule la chaine representant un L-System après un nombre (*nbItt*) ittérations.

Entrées : Le nombre d'ittération (*nbItt*) à faire
Sortie : La chaine issue de la production après *nbItt* ittération

```
1 etat ← axiom
2 pour i ← 0 à nbItt faire
3 |   etat = etatSuivant(etat)
4 fin
5 retourner etat
```

Calcul de la chaine interprétée en 2D et 3D

Algorithme 2 : Méthode **etatSuivant**

Entrées : etatActuel : chaine représentant l'état actuel
Sortie : Une chaine representant l'etat suivant

```
1  listedeRegleAleatoire ← construireListeDeRegle(regles)
2  pour chaque caractere dans etatActuel faire
3      trouver = Non
4      pour i ← 0 à taille de listedeRegleAleatoire faire
5          si debut de listedeRegleAleatoire(i) == caractere alors
6              trouver ← Oui
7              resultat+ = transformer listedeRegleAleatoire(i)
8              break
9          fin
10     fin
11     si trouver == Non alors
12         resultat+ = caractere
13     fin
14 fin
15 retourner resultat ;
```

Calcul de la chaine interprétée en 2D et 3D

Algorithme 3 : Méthode **construireListeDeRegle** : construit une liste de règle qui sera utilisée pour la production d'un état suivant.

Entrées : listeDeRegle : Une liste de règles
Sortie : Une autre liste de règles

```
1 listeDesDebutDesRegles ← obtenirDebutDesRegles(listeDeRegle)
2 pour caractere ← listeDesDebutDesRegles faire
3 |   regles ← regleCommencantPar(caractere, listeDeRegle)
4 |   resultat.ajouter(selectionnerUneRegleAléatoirement(regles))
5 fin
6 retourner resultat ;
```

Démonstration

Lancement de l'application

Exécutable .jar

1. cd dist/
2. java -jar Lsystem.jar

Script bash

1. `cd src && chmod 711 App.sh`
2. `./App.sh` pour lancer l'application

Démonstration

Problèmes rencontrés et solution proposées

Problèmes rencontrés et solutions proposées

Si on lance l'application et qu'on tente de générer un arbre avec un nombre d'itérations au-delà d'un certain seuil, la représentation 2D et 3D risque de déborder. Nous avons donc :

- Essayer de limiter la taille des chaines générées à 20000 caractères ;
- Ajouter la possibilité de dézoomer le rendu 3D.

Conclusion et perspectives

Conclusion

- La réalisation de ce projet sur l'interpréteur de systèmes de Lindenmeyer a été une expérience enrichissante et stimulante pour notre équipe
- En nous concentrant sur les objectifs définis, nous avons progressivement élaboré une application fonctionnelle et polyvalente, capable de générer des représentations visuelles à la fois en 2D et en 3D à partir de règles de réécriture de L-systèmes

Perspectives

- La simulation d'une foret par ajout de plusieurs arbres dans la même fenêtre cote à cote ;
- La possibilité de zommer et de dezommer le rendu 2D ;
- Faire en sorte que l'arbre grandisse dans la direction du soleil ;
- Ajouter des contrôles clavier sur le rendu 3D.