Projet *La fourmilière ou la simulation du dépot*

Le but est de développer une interface permettant de *simuler* le ramassage ou l'éparpillement de graines par des fourmis sur le terrain d'une fourmiliére.

1 Le noyau fonctionnel (ou modèle)

Une version de base du noyau fonctionnel *NoyauFourmiliere.zip* vous est fournie. Le paquet `terrain` contient l'essentiel, mais vous pourrez améliorer/modifier son contenu si besoin.

Le sol d'une fourmilière est constitué d'une grille $h \times l$ de cellules carrées. Chaque cellule peut contenir (ce n'est pas exclusif l'un de l'autre) : un mur, une fourmi, ou un nombre de graines inférieur à l'entier $qMax$. Une fourmi peut porter une seule graine.

A chaque seconde : une fourmi choisit ou non de prendre une graine si elle n'en porte pas déjà et si il y en a une sur la case où elle se trouve, la fourmi se déplace aléatoirement dans une des cases voisines, la fourmi choisit ou non de poser la graine si elle en porte une et s'il reste de la place sur la case. À chaque seconde de l'évolution de la fourmilière, toutes les fourmis agissent comme indiqué précédemment.

Le noyau fonctionnel fourni est minimal. Dans l'absolu, pour que la simulation soit plus intéressante, il faudrait -entre autres- améliorer la fonction qui compte les tas de graines (en comptant la taille des tas de graines qui se touchent, même si elles sont à distance de plus de deux cases) et la manière dont se déplacent les fourmis (telles qu'elles sont programmées, elles tournent un peu en rond) **mais je ne vous demande pas de programmer ces deux points**.

2 Interface

L'interface devra présenter tous les outils permettant de manipuler et faire évoluer cette fourmilière. Il y aura :

- le terrain ou plateau : initialement, ce sera un carré de 50×50 cellules, chaque cellule sera représentée par un carré de 10 pixels. Et la taille minimum du plateau sera de 200×200 pixels (soit 20×20 cellules). Une cellule est visualisée ainsi : un mur est noir, une cellule vide est blanche et plus ou moins rouge suivant le nombre de graines qu'elle contient (ou bien vous pouvez choisir un rond rouge plus ou moins gros suivant le nombre de graines). Une fourmi sera représentée par un rond vert ou bleu suivant si la fourmi porte une graine, le modèle fourni n'autorise la présence que d'une seule fourmi par cellule.
- un bouton *Loupe*,
- un bouton *Quit* en bas à droite de la fenêtre,
- un bouton *Play/Pause* qui lancera/arrêtera la simulation sur sur le plateau de jeu en changeant d'icône comme il le faut,
- des composants permettant d'afficher : le nombre de graines et de fourmis dans la fourmilière, le nombre d'itérations depuis le début de la simulation (ou le dernier reset),
- un ensemble de composants permettant de :
  - changer les paramètres : taille du plateau de jeu (qui sera toujours carré), capacité des cases, vitesse de la simulation avec un slider par exemple, etc
  - réinitialiser tout le plateau en le vidant,

– initialiser aléatoirement le plateau avec des probabilités de graines, de fourmis, et de murs choisies par l'utilisateur,

Pour la réinitialisation du plateau ou son changement de taille, il faudra ouvrir une fenêtre de confirmation.

• *Play/Pause* Initialement, l'interface est en mode *Pause* et le bouton *Play/Pause* permet de passer du mode *Pause* au mode *Play* et inversement.

Lorsque la simulation est en mode *Pause* : les composants permettant de modifier la fourmilière et ses propriétés sont actifs. Sur le plateau, la grille est dessinée (cela permet de se repérer). Il est possible de modifier le terrain de la fourmilière à la souris.

Lorsque la simulation est en mode *Play* : on ne peut modifier la fourmilière ni ses paramètres, mais on peut modifier la vitesse de passage au temps $t+1$ de manière à changer la rapidité de l'évolution. Quand la simulation est en mode *Play*, on ne dessinera pas la grille sur le plateau, seules les cellules et leur contenu seront affichées.

• *Bouton Loupe* le bouton loupe permet d'ouvrir/fermer une fenêtre 330x330 pixels dans laquelle on verra un zoom de 11x11 cellules centré sur la cellule pointée dans la fenêtre principale. Vous pourrez ici améliorer le dessin des graines et fourmis. Le bouton Loupe est actif que la simulation soit en Play ou en Pause.

• *Changement du terrain* Le changement du terrain se fait à la souris : un clic (ou drag) de souris avec le clic gauche permet de construire/détruire un mur ; un clic (ou drag) avec le clic gauche et touche shift enfoncée permet d'ajouter une fourmi ; une action sur la molette permet de changer le nombre de graines.
Remarque : dans le noyau fourni, on n'a pas le choix des fonctions de probabilité de prise ou dépôt de graine. Si vous le souhaitez et avez le temps, vous pourrez améliorer ce point.

3 Travail demandé

Ce projet doit être réalisé en binôme. Pour l'interface, vous ne devez utiliser que les classes standards de javaFX (pas d'autres composants tous faits, mais vous pouvez développer vos propre composants).

Le code doit être propre, modulaire, bien indenté, documenté. Vous utiliserez l'architecture MVC. L'ensemble des sources appartiendra au package `jeuDesFourmis`, plus précisément, le noyau fonctionnel dans le package `jeuDesFourmis.model`, et l'IHM dans le package `jeuDesFourmis.vue`, etc. Libre à vous d'ajouter autant de sous-packages que nécessaires.

Pour ce projet vous devrez :

- implémenter l'interface, vous devrez respecter l'agencement proposé pour les fenêtres, mais couleurs, décorations de boutons, et autres aspects esthétiques sont laissés à votre libre choix,
- gérer toutes les actions demandées par l'interface, boutons, souris, box, et zones de saisie diverses,
- gérer les fenêtres de confirmation pour les 3 manipulations effaçant ou écrasant des données dans le plateau de jeu.

Attention, vos fenêtres doivent pouvoir être redimensionnées jusqu'à un certain point sans que cela ne provoque de «catastrophe». Le code doit être aussi «propre» que possible : pas de redondance de code, un code lisible, modulaire (par exemple, changer la dimension de la zone tampon ou la taille par défaut du plateau de jeu ne doit pas prendre plus de 10 secondes), et commenté.

Vous rendrez un rapport d'au plus 10 pages, au format pdf (**à l'exclusion de tout autre**), expliquant comment est structurée votre application, comment vous avez résolu les problèmes de conception

rencontrés, et comment votre application peut être étendue pour produire l'application rêvée et vous indiquerez comment vous vous êtes répartis le travail dans le binome.

Votre travail sera rendu sur UPdago dans une archive `nom1_nom2.tar.tgz`, où `nom1` et `nom2` sont évidemment vos noms. À l'intérieur de cette archive, on trouvera le répertoire `nom1_nom2` qui contiendra le rapport et le package `jeuDesFourmis` contenant votre code source.

Nous nous réservons le droit de convoquer certains binômes à un oral pour d'éventuelles précisions sur le travail rendu. Il est bien entendu qu'on attend un travail original (c'est-à-dire personnel, tout plagiat serait très fortement pénalisé), et équitablement réparti au sein de chaque binôme.

4 Évaluation

Les critères pris en compte pour la notation :

- le code rendu compile et le programme peut être lancé et manipulé en particulier **le code doit être indépendant de toute IDE, et compilable en ligne de commande**, vous devrez indiquer dans votre rapport ou dans un README.txt la commande permettant de compiler et exécuter votre code ;
- les différentes consignes sont respectées (structuration et format de l'archive rendue, etc.) ;
- le code est propre, modulaire, respectant l'architecture MVC, clair et lisible, documenté... ;
- l'application est ergonomique ;
- si vous avez repéré des erreurs dans votre application, sans avoir le temps de les corriger, elles sont documentées dans le rapport;
- le rapport donne les informations nécessaires pour comprendre votre démarche, et utiliser votre application.