

Module Java Avancé

AntHill

Documentation utilisateur



Responsable : Rémi Forax

Etudiants : Adrien Spanier et Stéphane Jacquemain

Sommaire

I. Introduction	2
II. Présentation d'une carte	3
III. Afficher les statistiques	5
a) Déroulement de la simulation	5
b) Etat d'une case.....	5
IV. Utilisation du script ant	6
V. Lancement de l'application.....	7
VI. Mise en place d'une carte.....	8
c) Configuration d'une carte.....	8
d) Carte aléatoire	9
e) Carte manuelle.....	10

I. Introduction

Ce manuel utilisateur détaille l'utilisation du package antHill.jar. Il vous permettra de créer une carte avec les paramètres de votre choix et ensuite de la charger.

Il est également expliqué comment charger son propre dieu des fourmis.

II. Présentation d'une carte

Exemple de carte générée aléatoirement.



Il y a deux types d'insectes. Les fourmis et les tueurs de fourmis.



La fourmi travailleuse est celle qui va chercher de la nourriture sur des zones **REGENERATE_FOOD** et qui la régurgite sur les zones où la reine peut pondre. La quantité maximale transportable est réglable ainsi que le temps nécessaire pour manger et régurgiter.



La reine récupère la nourriture et peut ainsi pondre. La quantité nécessaire pour pondre est également paramétrable.



Lorsqu'un tueur rencontre une fourmi, il lui enlève de la vie. La quantité de vie est modifiable.

Voici une définition pour les 4 types de cases possibles avec la valeur correspondante dans le fichier de configuration ainsi que l'image associée.

➤ Case normale

- NORMAL



➤ Obstacle, case ou aucun insecte ne peut se rendre.

- BLOCKING



➤ Endroit ou la reine a le droit de pondre.

- LAYABLE



➤ Endroit ou est positionnée la nourriture. Lorsque la nourriture est épuisée, elle est régénérée d'une certaine quantité après un certain temps défini en configuration.

- REGENERATE_FOOD



La quantité de nourriture est représentée par 3 images différentes suivant la quantité restante.

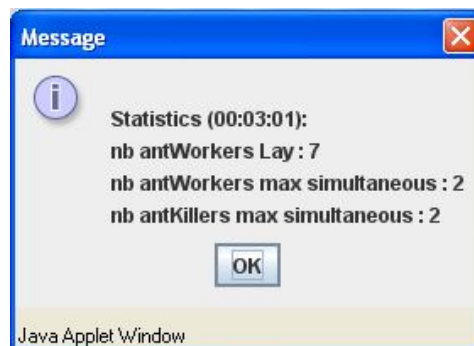


Le temps pendant lequel une fourmi perd de la vie est matérialisée par une croix rouge. La croix disparaît dès que la fourmi n'est plus en contact avec un tueur. La quantité de vie perdue est proportionnelle à la durée du contact avec les tueurs. Elle est réglable par milliseconde dans le fichier de configuration.



III. Afficher les statistiques

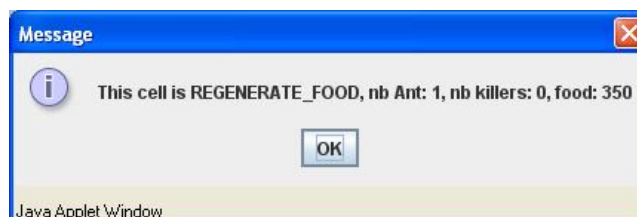
a) Déroulement de la simulation



Des statistiques sont accessibles à tout moment en appuyant sur la touche 'S' donnant des informations sur :

- la durée de vie de la fourmilière (i.e. de la reine);
- le nombre de travailleuses créées depuis le début de la simulation;
- le nombre maximum de travailleuses simultanées dans la fourmilière;
- le nombre maximum de tueuses simultanées dans la fourmilière.

b) Etat d'une case



Des statistiques individuelles sur les cases de la fourmilière peuvent également être données en cliquant sur une case et donnant des informations sur :

- le type de la case;
- la quantité de nourriture présente sur cette case;
- le nombre de fourmis présentes sur cette case (reine incluse);
- le nombre de tueuses présentes sur cette case.

IV. Utilisation du script ant

Un script ant est fourni et permet :

- La compilation des sources
 - ant compileCompilation des sources du dossier src dans le dossier classes.
- La création de antHill.jar
 - ant jarCette commande crée antHill.jar avec les **.class** du dossier classes, inclus les ressources dans antHill.jar, crée un dossier lib avec les librairies nécessaires et un dossier gota inclus par défaut dans le Class-Path du manifest . Tous les fichiers créés sont ajoutés dans un dossier dist.
- La génération de la javadoc
 - ant javadoc
- Le nettoyage du projet
 - ant clean

Le script ant est le fichier build.xml. Ce fichier utilise un fichier **.properties** nommé **build.properties** dans lequel est inscrit le nom du jar, du dossier source, etc.

V. Lancement de l'application

Pour lancer l'application il suffit de se placer dans le dossier correspondant et d'exécuter la commande suivante :

java -jar antHill.jar fichierDeConfiguration.xml

(Équivalent à : `java -jar antHill.jar fichierDeConfiguration.xml fr.umlv.anthill.gota.GotA`)

Avec la commande précédente, un dieu est chargé par défaut. Pour charger un autre dieu, il est nécessaire de donner le nom de la classe en deuxième paramètre comme suit :

java -jar antHill.jar fichierDeConfiguration.xml nouveauDieu

Lorsque le script ant est exécuté avec la commande: `ant jar`, un dossier `gotA` est créé. Le `nouveauDieu` doit être placé dans ce dossier. Si vous souhaitez charger un `gotA` d'un autre dossier, il faudra ajouter son chemin dans le Class-Path du fichier manifest de `antHill.jar`.

Pour que le `gotA` soit valide, il doit implémenter l'interface suivante :

fr.umlv.anthill.simulator.GotAInterface

VI. Mise en place d'une carte

c) Configuration d'une carte

La carte est créée à partir d'un fichier xml qui doit respecter une des deux configurations possibles.

Toute carte est définie entre deux balises : `<map>` `</map>`

Une partie de la configuration est commune, ce sont les options de configuration du simulateur et de la fourmilière. La configuration d'une option se fait entre les balises :

`<entry id=nomDeL'Option>valeurDesirée</entry>`

Liste des options disponibles:

- **antWorkersMaxCount** : nombre maximum de travailleuses sur la carte;
- **antKillersMaxCount** : nombre maximum de tueurs sur la carte;
- **movingDuration** : durée du mouvement d'un insecte (en ms);
- **eatingDuration** : durée d'acquisition d'un point de nourriture par une travailleuse;
- **layingDuration** : durée de la ponte de la reine;
- **releasingFoodDuration** : durée de dépôt d'un point de nourriture;
- **antWorkerHealthMax** : points de vie initiaux d'une travailleuse;
- **antQueenHealthMax** : points de vie initiaux de la reine;
- **regeneratingFoodDuration** : durée de régénération d'une case de nourriture;
- **regeneratingFoodAmount** : quantité régénérée par une case de nourriture;
- **foodNeededToLay** : quantité de nourriture nécessaire à la reine pour pondre;
- **antWorkerFoodMax** : quantité maximum de nourriture transportée par une travailleuse;
- **antQueenFoodMax** : quantité maximum de nourriture transportée par la reine;
- **percentageOfAntKillersPerAntWorkers** : proportion de tueurs en fonction du nombre de travailleuses (en %);
- **healthLostPerMs** : nombre de points de vie perdus par une fourmi par milliseconde;
- **antHillName** : nom de la fourmilière;
- **cellSize** : taille d'une cellule (en pixel);
- **mapWidth** : largeur de la carte (en nombre de cellules);
- **mapHeight** : hauteur de la carte (en nombre de cellules);
- **displayUpdateFrequency** : fréquence d'affichage (en ms).

L'utilisation de ces options est facultative, le jeu utilisant des valeurs par défaut.

La suite du fichier concerne la fourmilière en elle-même. Il existe deux types de configuration de carte : aléatoire ou normale. Une configuration aléatoire permet de définir un minimum d'options permettant de générer une fourmilière aléatoire. Une configuration normale permet de définir précisément tous les éléments créés au démarrage de la carte.

d) Carte aléatoire

Exemple d'un fichier XML d'une carte aléatoire.

```
<map type="random">// Ici on indique que le type de la carte sera aléatoire.

  // Deux options sont personnalisées.
  <entry id="antWorkersMaxCount" value="20" />
  <entry id="antKillersMaxCount" value="20" />

  // Largeur et hauteur de la carte en nombre de cases.
  <cells x="20" y="10">

    // Ajout d'un nombre fixé (5) de cases du type REGENERATE_FOOD.
    <cell count="5" type="REGENERATE_FOOD" />

    // Au moins une case LAYABLE doit être définie
    <cell count="15" type="LAYABLE" />
    <cell count="25" type="BLOCKING" />

    // Ajout d'un nombre de cases infini (0) de type NORMAL.
    // Permet de remplir aisément la carte dans sa totalité.
    // Attention une seul zone infini peut être définit
    <cell count="0" type="NORMAL" />

  </cells>

  // Nombre de travailleuses au démarrage du simulateur.
  <antWorker count="2" />
</map>
```

e) Carte manuelle

Exemple d'un fichier XML d'une carte manuelle.

```
<map type="manual"> // Ici on indique que le type de la carte sera manuelle.

    <cells x="4" y="5"> // Comme en aléatoire, on définit la largeur et la hauteur de la carte.

        // Déclaration des cases
        // Notez que les coordonnées des cases commencent à 0, qu'une coordonnée ne
        // peut être utilisé qu'une fois et que le nombre de cases doit être égal à (hauteur *
        // largeur).

        // Les cases LAYABLE doivent être adjacentes.
        <cell x="0" y="0" type="LAYABLE" />
        <cell x="0" y="1" type="LAYABLE" />
        <cell x="0" y="2" type="LAYABLE" />
        <cell x="0" y="3" type="LAYABLE" />
        <cell x="0" y="4" type="LAYABLE" />
        <cell x="1" y="0" type="NORMAL" />
        <cell x="1" y="1" type="BLOCKING" />
        <cell x="1" y="2" type="NORMAL" />
        <cell x="1" y="3" type="BLOCKING" />
        <cell x="1" y="4" type="NORMAL" />
        <cell x="2" y="0" type="NORMAL" />
        <cell x="2" y="1" type="NORMAL" />
        <cell x="2" y="2" type="NORMAL" />
        <cell x="2" y="3" type="NORMAL" />
        <cell x="2" y="4" type="REGENERATE_FOOD" />
        <cell x="3" y="0" type="NORMAL" />
        <cell x="3" y="1" type="NORMAL" />
        <cell x="3" y="2" type="REGENERATE_FOOD" />
        <cell x="3" y="3" type="BLOCKING" />
        <cell x="3" y="4" type="NORMAL" />
    </cells>

    // Création de travailleuse au démarrage.
    // Les travailleuse ne peuvent pas se situer sur une case BLOCKING.
    <antWorker x="0" y="0" />
    <antWorker x="2" y="2" />
</map>
```

Pour les deux types de cartes, la présence d'une case de pont sur la carte est obligatoire.