Systèmes Multi-Agents
–
Mise en œuvre à l'aide de MaDKit

1 Prise en main de MaDKit ~25'

Si vous travaillez sur votre ordinateur personnel, vous pouvez travailler sous Linux ou Windows, selon votre préférence. L'important est de disposer du JDK et d'un IDE Java. Sur les machines de l'école, seul le boot Linux est doté du JDK et d'Eclipse.

1.1 Installation

La simulation de systèmes multi-agents se fera à l'aide de MaDKit.

Téléchargez MaDKit puis décompressez le dans votre compte. *Et voilà*.

Au passage parcourez rapidement le site de MaDKit pour vous faire une idée générale de ses fonctionnalités.

1.2 Premiers lancements

Pour vérifier que tout est fonctionnel, placez-vous dans le répertoire résultant de la décompression de MaDKit puis lancez dans un terminal la commande :

```
$ java -jar madkit-5.3.2.jar
```

À ce stade vous devez voir apparaître une interface graphique, le "desktop" MaDKit. À partir de cette interface vous pouvez lancer quelques simulations simples.

Utilisez le menu "`MaDKit` → `Load jar file`" pour charger le jar d'une des démonstrations. Ces démonstrations se trouvent dans le répertoire "demo" de l'archive décompressée. Une fois que vous avez chargé le jar d'une démo, des entrées apparaissent dans les menus de MaDKit.

Vous pouvez explorer quelques minutes l'exécution d'une de ces démos (`bees` par exemple).

Cette interface graphique est utile pour les utilisateurs de simulations multi-agents déjà définies : les personnes qui vont explorer les paramètres de la simulation et en exploiter les résultats. Cependant l'objectif de cette mise en pratique est de vous amener à **définir** une simulation multi-agent, c'est-à-dire à définir les **comportements** individuels des agents et leurs **interactions**, entre eux et avec leur environnement. Lors de cette phase de conception et d'implémentation d'une simulation il est souhaitable de pouvoir lancer une simulation à partir d'un IDE ou d'une ligne de commande.

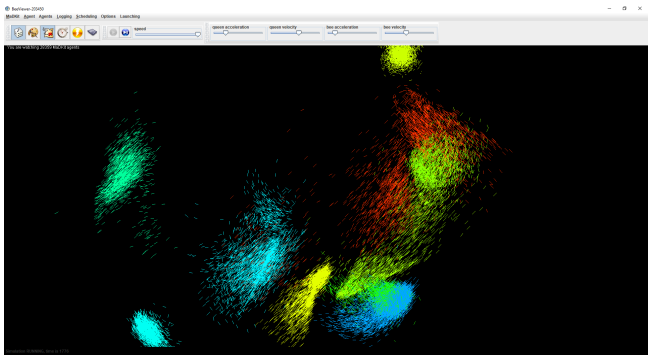

FIGURE 1 – Simulation Bees de MaDKit

1.3 Depuis la ligne de commande

MaDKit offre la possibilité de lancer une simulation facilement depuis la ligne de commande. Dans le répertoire "demos" résultant de la décompression de l'archive, exécutez la commande :

```
$ java -jar MaDKit-bees-2.0.0.8.jar
```

Vous devez voir apparaître une instance de MaDKit et l'exécution de la démonstration "bees" devrait se lancer ce qui devrait donner quelque chose comme la Figure 1. Si ce n'est pas le cas, lisez attentivement les messages d'erreur, essayez de résoudre les erreurs indiquées et seulement en cas de problème persistant.

2 Premier agent ~20'

L'objectif de cette section est de vous faire développer votre premier agent. Pour cela, dans votre IDE préféré (Eclipse, IntelliJ, vim), créez un nouveau *projet* vide Java. Dans ce projet, ajoutez comme bibliothèque externe, le fichier `jar` de MaDKit (normalement `madkit-5.3.2.jar`).

Créez un nouveau *fichier* Java vide nommé `AgentLifeCycle.java` et copiez dans ce nouveau fichier la classe décrite dans le tutoriel "helloworld→ex02→AgentLifeCycle" MaDKit que vous trouverez là : www.madkit.net/madkit/tutorials/html/hello_world/index.html

Une fois votre fichier complété avec la définition de la classe, lancez le **`main`** de cette classe à partir de votre IDE. Vous devriez voir se lancer le programme décrit dans le tutoriel.

Si vous le désirez, vous pouvez bien sur lancer votre agent depuis une ligne de commande en ajoutant le "jar" de MaDKit au `CLASSPATH` lors de la compilation et du lancement de votre agent. Cela donnera à quelque chose comme ça pour le lancement :

```
$ java -cp /chemin/vers/madkit-5.3.2.jar:./ votre.package.AgentLifeCycle
```

Ne **passez pas** à l'étape suivante tant que cette première version simple ne se lance et ne s'exécute pas correctement.

3 Premières communications ~25'

Afin de bien comprendre les méthodes de communication entre agents offertes par MaDKit, faites fonctionner les classes des tutoriels **1 à 4** de la partie communication du tutoriel que vous trouverez là : www.madkit.net/madkit/tutorials/html/communication/index.html

Remarquez :

- La façon dont un agent envoie un message.
- La façon dont les agents qui doivent être lancés au démarrage sont précisés.
- La façon dont les rôles sont attribués aux agents puis utilisés dans la communication.

4 Exercice d'application - Jeux de rôles ~30'

Votre compte-rendu de TP devra comporter, entre autres, des éléments d'information sur cette partie. Consultez la partie 7 Évaluation pour les détails.

L'objectif de ce premier exercice d'application est de travailler avec des rôles dynamiques.

Vous devez réaliser trois types d'agents :

- Des agents émetteurs, en charge de lancer des messages de manière aléatoire.
- Des agents compteurs, en charge de compter le nombre de message reçus.
- Des agents contrôleurs, qui doivent surveiller qu'un nombre suffisant d'agents compteurs sont présents dans le système.

Pour les émetteurs, vous reprendrez un des agents des tutoriels que vous avez fait fonctionner. Leur comportement consiste à envoyer un nombre aléatoire de messages aux agents compteurs, puis à se terminer.

Les compteurs, comme leur nom l'indique, sont en charge de compter le nombre de messages qu'ils reçoivent depuis les agents émetteurs. Au bout d'un certain temps, aléatoire, un agent compteur se "transforme" en agent émetteur. Avant de se transformer, l'agent compteur doit envoyer à un agent contrôleur un message indiquant le nombre de messages qu'il a reçu jusqu'ici. Pour "transformer" un agent compteur en émetteur, utilisez la méthode **`launchAgent`** qui permet de lancer un agent à partir d'un autre.

Les agents contrôleurs doivent s'assurer que le nombre d'agents compteurs est constant. Pour cela ils doivent créer un agent compteur lorsque l'un des compteurs indique qu'il est sur le point de disparaître. Le *nouvel* agent compteur doit démarrer avec un nombre de messages égal à celui indiqué par l'agent compteur qu'il remplace.

5 Adaptation de *Bees* ~25'

Si vous voulez avoir un aperçu de ce que donne la démonstration "Bees", placez vous dans le répertoire demos puis lancez la : `java -jar MaDKit-bees-2.0.0.8.jar`

Vous pouvez jouer un peu avec les paramètres et observer le comportement des "abeilles".

5.1 Code source de *Bees*

L'objectif de cette partie étant d'adapter cette démonstration, vous allez créer un nouveau projet java incluant ses sources.

Le **code** de la simulation "Bees" est inclus dans l'archive de MaDKit, dans le répertoire "demos", dans l'archive : `MaDKit-bees-2.0.0.8-src.zip` (l'archive zip, pas le jar)

Après avoir décompressé cette archive, ajoutez le code source des classes de cette simulation dans un nouveau projet Java vide. Prenez soin d'ajouter la dépendance à la bibliothèque MaDKit en ajoutant là encore le fichier jar de MaDKit.

Assurez vous que la simulation compile et s'exécute lorsque vous lancez le main de la classe `BeeLauncher`. Il vous faudra notamment **adapter** les déclaration de package des classes.

5.2 Modifications

Pour prendre en main cette simulation, modifiez la :

1. Faites en sorte que les *reines* soient représentées par des cercles et non plus comme toutes les autres abeilles par des lignes.
   - Dans le rendu graphique des abeilles (méthode `computeFromInfoProbe` de la classe `BeeViewer`), vous devrez déterminer si l'abeille en train d'être dessinée est une simple abeille ou une reine. Pour cela, vous pouvez ajouter une propriété à la classe `BeeInformation` ou utiliser l'opérateur `instanceof` de Java.
   - Vous utiliserez la fonction `fillOval` du package `Graphics` de AWT pour tracer les reines.
2. Modifiez les paramètres de génération et de destruction des abeilles et des reines pour qu'il soit plus facile d'identifier le comportement des abeilles. Vous pouvez par exemple :
   - Générer moins d'abeilles lorsqu'une génération d'abeilles est choisie aléatoirement.
   - Augmenter la durée de vie des reines.

Vous pouvez à ce stade apporter des modifications qui vous intéressent à la simulation, sans toutefois modifier le comportement des abeilles. Ceci est l'objet du prochain exercice.

Une modification intéressante pour faire des captures d'écran pour votre rapport et votre soutenance consiste à mettre un fond clair à la simulation.

6 Proies prédateurs

Dans cet exercice, vous devez modifier le comportement des abeilles afin de simuler le comportement d'une ruche face à une attaque de frelons asiatiques. Les différents comportements sont les suivants :

- Un frelon chasse les abeilles qui sont à proximité.
- Si une abeille se trouve à une distance trop faible d'un frelon, celui-ci la bloque et la tue en 3 secondes.
- Si un frelon se retrouve entouré (à faible distance) de plus de 7 abeilles, il meurt en 5 secondes (comportement observé dans la nature).
- Un frelon ne peut plus attaquer une abeille s'il est entouré de plus de 4 abeilles.

Programmez les comportements des abeilles et des frelons selon ces conditions. Procédez par étape : par exemple validez d'abord le fait qu'un frelon bloque et tue une abeille lorsqu'il passe à proximité. Validez ensuite que la présence d'autres abeilles empêche le frelon de tuer une abeille.

Vous pouvez :

- modifier les temps de survie et les nombres d'abeilles des différentes conditions ;
- laisser libre cours à votre imagination pour la mise en place de stratégies d'attaque et de défense, basées sur la communication et/ou la perception.

N'hésitez pas à consulter les tutoriels et la documentation de MaDKit si vous rencontrez des difficultés.

7 Évaluation

L'évaluation porte sur deux éléments, un compte-rendu de TP et une soutenance.

7.1 Compte-rendu

Le compte-rendu du TP devra présenter rapidement l'outils MaDKit, son utilité et son fonctionnement. Le document devra de plus présenter de manière synthétique le travail réalisé dans les parties 4, 5 et 6.

Pour chaque partie indiquez de manière synthétique :

- votre niveau d'avancement ;
- les éventuelles difficultés rencontrées ;
- les solutions mises en œuvre ;
- les éléments d'apprentissage que vous avez abordés au cours du travail.

Des extraits judicieux de code, des diagrammes de classe et des captures d'écran sont les bien venues.

7.2 Soutenance - 10 minutes + 10 minutes Q&R

Lors de la soutenance, vous devrez présenter les éléments pertinents de votre compte-rendu, là encore le but est de mettre en valeur votre travail et votre compréhension.

Pour les téméraires, des démonstrations peuvent être réalisées avec les risques que cela comporte sur la qualité de la présentation. Les vidéos sont une solution plus sure mais plus onéreuse en temps. Dans tous les cas cette partie démonstration ne devra pas excéder 3 minutes sur vos 10 minutes de présentation.