Java avancé – 3A

Projet annuel

2013-2014

Projet annuel

Simulation du fourragement dans une colonie de fourmis

1. Description du projet

Ce projet vise à étudier le comportement d'une colonie de fourmis dans leur activité de fourragement (qui consiste à aller chercher des ressources dans l'environnement et les ramener à la fourmilière). Les fourmis sont des organismes qui ont des comportements individuels élémentaires, mais dont les groupes montrent des comportements émergents d'une stupéfiante organisation. C'est le cas par exemple de la « ligne » de fourmis que l'on observe fréquemment dans la nature.

http://fr.wikipedia.org/wiki/Algorithme_de_colonies_de_fourmis

Le but du projet est de construire un simulateur multi-agent représentant une colonie de fourmis dans ses activités de fourragement et analyser l'impact de divers paramétrages[1] de la simulation sur l'efficacité du fourragement (mesuré par un taux dont le calcul est à définir).

Côté algorithmes, vous pouvez vous documenter sur les *Ant Colony Optimisazion* (ACO). L'idée générale est de comprendre comment des règles simples au niveau individuel peuvent faire émerger

[1] Entre d'autres : nombres de fourmis, taille du monde, quantité de nourriture disponible, dispersion de la nourriture dans l'environnement, vitesse d'évaporation des phéromones, présence d'obstacles entre la fourmilière et la nourriture, dispersion des obstacles, stratégies d'évitement des obstacles …

des comportements collectifs « complexes ». Cette complexité est souvent le fait d'une grande variabilité des comportements collectifs, produit du nombre élevé d'interactions fourmi-fourmi et fourmi-environnement. Ces solutions algorithmiques sont parfois retenues face au problème de l'animation vraisemblable des personnages non joueurs dans les jeux vidéo, ou dans le routage des réseaux informatiques.

Règles de comportement des fourmis et évolution de l'environnement

- une fourmi localisée au même endroit que la nourriture et qui n'en porte pas déjà en prélève.
- une fourmi qui porte de la nourriture sait rentrer intuitivement à la fourmilière (sans pour autant disposer d'une carte de l'environnement).
- une fourmi qui porte de la nourriture sécrète des phéromones[2] qu'elle dépose où elle se trouve. Ces phéromones s'évaporent avec le temps et finissent par disparaître si d'autres phéromones ne sont pas déposées au même endroit. Si l'environnement compte déjà une pastille de phéromones à cet endroit, leurs quantités s'ajoutent.
- une fourmi qui porte de la nourriture et qui se trouve sur la fourmilière la dépose et repart en chercher. Elle n'a pas en mémoire la localisation de la dernière source de nourriture.
- une fourmi qui ne porte pas de nourriture et qui ne se trouve pas sur une source de nourriture se déplace :
    - aléatoirement sur les positions phéromonées du voisinage si celui-ci en contient
    - aléatoirement sur toutes les positions du voisinage sinon.
- en cas de présence d'obstacles, les fourmis ne peuvent mettre en œuvre que des stratégies locales d'évitement. Les comportements ci-dessus doivent être adaptés en conséquence.

Pour modéliser la notion de voisinage, on pourra choisir (entre autres) :

- le voisinage classique entre pixels (1 position = 1 pixel). Il y a donc 8 positions dans le voisinage d'une position et 8 déplacements possibles *(N, NE, E, SE, S, SO, O, NO).*
- le voisinage d'une grille d'hex (1 position = 1 hexagone). Dans ce cas, il n'y a que 6 voisins mais la distance réelle entre une position et un de ses voisins est plus homogène, et donc plus réaliste.

La simulation se fera pas à pas : les fourmis se déplacent dans une position de leur voisinage, et les phéromones s'évaporent.

[2] Des substances chimiques utilisés pour communiquer via l'environnement : http://fr.wikipedia.org/wiki/Ph%C3%A9romone

2. Spécifications

Version sans extension (=12/20)

L'interface de votre application est à coder en Swing. L'interaction avec l'utilisateur est la suivante :

1. Dans une boîte de dialogue personnalisée, l'utilisateur est invité à saisir les différents paramètres de sa simulation. Les champs contiennent tous des valeurs par défaut. L'utilisateur a la possibilité de sauvegarder et de charger des configurations de paramètres. La boîte de dialogue contient un bouton qui lance la simulation dans une autre fenêtre.
2. La fenêtre de simulation montre l'évolution pas à pas d'un environnement comprenant un ensemble de fourmis, des phéromones, des sources de nourriture, une fourmilière, ainsi que des informations relatives à la simulation figurant sur un coin de la fenêtre.
    a. Les fourmis seront représentées par de très petits cercles noirs, avec au centre une pastille plus petite et de couleur différente si elles transportent de la nourriture.
    b. L'aspect visuel des sources de nourriture renseigne sur la quantité de nourriture en présence.
    c. la fourmilière est représentée par un carré plein
    d. les phéromones ont l'aspect de petits points gris. L'intensité de leur couleur pourra, ou non, varier selon la quantité de phéromone.
    e. les informations relatives à la simulation pourront comprendre : le numéro du pas de simulation, la quantité de nourriture rapportée à la fourmilière, la quantité restante dans l'environnement, la quantité totale de phéromones présente dans l'environnement.

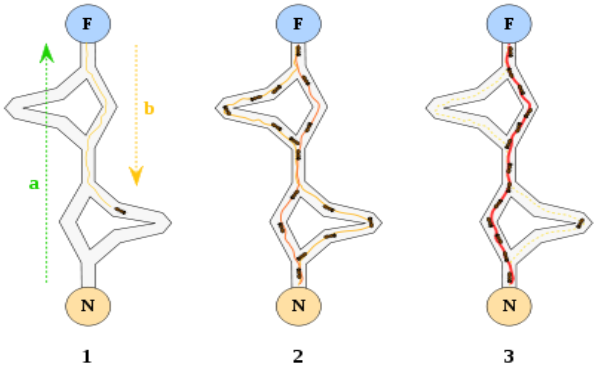
Autant que possible, on veillera à doter les fourmis de comportements réalistes. Une fourmi qui se déplace aléatoirement ne bourdonne pas. De manière générale, il sera bon d'introduire de l'aléatoire dans les autres mouvements (lorsqu'elles rentrent à la fourmilière par exemple).

Version avec extensions (>12/20)

Les extensions possibles sont (mais ne se limitent pas à) :

1. l'adoption d'une architecture MVC
2. la gestion d'obstacles à l'aide de stratégies locales d'évitement. L'aspect intéressant de ce point est de pouvoir faire émerger la faculté d'optimisation du système, c'est-à-dire la capacité à déterminer un chemin plus court parmi plusieurs chemins possibles[3]. Pensez dans ce cas à inclure la spécification d'obstacles dans les paramètres de la simulation, afin de pouvoir définir à la main des configurations « intéressantes » d'obstacles. (les coder en dur sinon) ; A noter que le type d'obstacle (circulaire, rectangulaire, convexe, à rebroussement, …) est souvent déterminant pour l'efficacité de la stratégie d'évitement. Veillez à bien identifier les limites de votre solution.

[3] http://fr.wikipedia.org/wiki/Algorithme_de_colonies_de_fourmis



Une configuration intéressante d'obstacle

Contraintes

L'architecture de votre application doit impérativement comprendre des classes *Simulation*, *Rendu*, et *Controleur*.

La classe *Simulation* comprend les éléments simulés (monde, fourmis, phéromones et fourmilière), et possède une méthode *void nextStep()* permettant de faire avancer la simulation d'un pas.

La classe *Rendu* se charge de l'affichage des éléments simulés et possède une méthode void paint(Simulation sim) qui rafraîchit l'affichage en prenant en compte une nouvelle configuration de la simulation.

La classe *Controleur* qui coordonne les 2 précédentes. Le Thread.sleep(…) sert à ralentir la vitesse de l'animation. **Votre code doit impérativement comprendre les lignes suivantes** :

```
…
Simulation sim = new Simulation(…);
Rendu rendu = new Rendu(…) ;
…
…
while (true)  {
      sim.nextStep();
      rendu.paint(sim);
      try {
            Thread.sleep(SleepDuration);
      } catch (InterruptedException e) {
            e.printStackTrace();
}
```

Notation

Outre l'adéquation au cahier des charges, le respect des contraintes et le nombre d'extensions réalisées, la notation prendra en compte

1. la faculté de montée en charge. Vous devez être capables de justifier le choix de vos collections d'objets, et avoir une idée de la charge maximale acceptée par votre application.
2. la lisibilité du code, l'absence de code dupliqué, la juste dose de commentaires
3. sa capacité à intégrer de nouvelles fonctionnalités

Informations pratiques

Le projet doit être fait en trinôme, dont la composition doit être décide et annoncée par email avant le 15 mai 2014. L'entraide et les discussions entre trinômes sont autorisées, voire encouragées, sous certaines conditions à la fois strictes et de bon sens: avoir déjà réfléchi à la structure de votre code auparavant et coder soi-même le projet

- **Présoutenances** : en cours, le mardi 3 juin 2014. Préparer différents paramétrages de votre simulation pour montrer l'impact de ces configurations sur l'efficacité du fourragement.

- **Rendu** : Le projet devra être envoyé par email à fbaudoin@my-itcampus.com avant le dimanche 22 juin 2014 à minuit. Vous joindrez à votre code une description
    - des fonctionnalités que vous avez implémentées
    - des difficultés rencontrées
    - des limitations de votre solution (surtout dans le cas d'obstacles)
    - des éventuels bugs.

- **Soutenances** : Elles auront lieu le mercredi 25 juin 2014. Préparez des paramétrages préenregistrés pour montrer la capacité de vos fourmis à contourner des obstacles. Après votre présentation, il vous sera demandé de modifier votre code de manière à intégrer des spécifications supplémentaires.