Projet L1, S2, 2018 : Simulation de fourmis

1 Introduction

On considère une grille de 20 lignes 20 colonnes (mais qui pourra être plus grande). Une case de la grille peut être vide, ou contenir une des trois choses suivantes : une fourmi, du sucre ou un élément de nid. Une case ne peut pas contenir en même temps deux fourmis, ou bien du sucre et une fourmi (sauf si la fourmi porte le sucre), ou bien un élément de nid et une fourmi. Les fourmis doivent donc contourner le sucre et le nid, et se contourner entre elles. Une fourmi peut ramasser du sucre si elle se trouve à coté d'une case en contenant et peut poser du sucre dans le nid si elle en porte et se trouve à coté d'une case contenant un élément de nid.

Au début, les fourmis cherchent du sucre aléatoirement sur la grille. Une fois qu'une fourmi a trouvé du sucre, elle le ramène vers le nid en suivant des "phéromones de nid", qui indiquent le chemin de retour vers le nid. Pendant leur retour du tas de sucre vers le nid, les fourmis marquent leur chemin avec des "phéromones de sucre", qui vont permettre aux autres fourmis de trouver le sucre plus rapidement. Les phéromones sont des substances émises par la plupart des animaux et certains végétaux, et qui agissent comme des messagers sur des individus de la même espèce. Extrêmement actives, elles agissent en quantités infinitésimales, si bien qu'elles peuvent être détectées, ou même transportées, à plusieurs kilomètres.

Pour simplifier, nous supposerons que les "phéromones de nid" sont présentes en permanence et installent un gradient dirigé vers le nid. Cela signifie que l'intensité des phéromones augmente au fur et à mesure que l'on se rapproche du nid. Pour se rapprocher du nid, il suffit donc de se déplacer vers une case de plus forte intensité en phéromones. Les "phéromones de sucre", elles, s'évaporent. Ainsi, les fourmis ne s'obstineront pas à suivre un chemin de phéromones de sucre vers un tas de sucre qui a été complètement vidé.

1.1 Comportement des fourmis

Pour décrire le comportement des fourmis, nous utiliserons les prédicats (fonctions booléennes) listés ci-dessous.

- Prédicats sur une fourmi f :

`chercheSucre(`f`)`	f ne porte pas de sucre
`rentreNid(`f`)`	f porte du sucre

- Prédicats sur une place (une case) p :

`contientSucre(`p`)`	p contient du sucre
`contientNid(`p`)`	p contient un élément de nid
`contientFourmi(`p`)`	p contient une fourmi
`vide(`p`)`	p ne contient ni sucre, ni fourmi, ni élément de nid
`surUnePiste(`p`)`	l'intensité des phéromones de sucre sur p est non nulle

- Prédicats sur une place p_1 et une place voisine p_2 :

`plusProcheNid(`p_1`, `p_2`)`	l'intensité des phéromones de nid sur p_1 est plus importante que celle sur p_2
`plusLoinNid(`p_1`, `p_2`)`	l'intensité des phéromones de nid sur p_1 est moins importante que celle sur p_2

Pour une place donnée, plusieurs prédicats peuvent être vrais, par exemple : `surUnePiste(`p`)` et `vide(`p`)`.

Le principe général du comportement des fourmis est le suivant. Elles cherchent du sucre en bougeant aléatoirement. Quand elles trouvent de la phéromone de sucre, elles s'éloignent le plus vite possible du nid, tout en restant sur la piste tracée par la phéromone de sucre. Quand elles sont à côté du sucre, elles chargent une partie du sucre, puis se rapprochent le plus vite possible du nid tout en posant des phéromones de sucre sur les cases empruntées.

Plus formellement, le comportement d'une fourmi est dicté par les règles énoncées ci-dessous par **ordre de priorité décroissante**.

Règles :

Soient f une fourmi, p_1 la case qu'elle occupe et p_2 une case adjacente à p_1.

1. Si `chercheSucre(`f`)` et `contientSucre(`p_2`)`,
   alors f charge du sucre et pose une phéromone de sucre sur p_1.
2. Si `rentreNid(`f`)` et `contientNid(`p_2`)`,
   alors f pose son sucre.
3. Si `rentreNid(`f`)` et `vide(`p_2`)` et `plusProcheNid(`p_2, p_1`)`,
   alors f se déplace sur p_2 et pose une phéromone de sucre sur p_2.
4. Si `chercheSucre(`f`)` et `surUnePiste(`p_1`)` et `vide(`p_2`)` et `plusLoinNid(`p_2, p_1`)` et `surUnePiste(`p_2`)`,
   alors f se déplace sur p_2.
5. Si `chercheSucre(`f`)` et `surUnePiste(`p_2`)` et `vide(`p_2`)`,
   alors f se déplace sur p_2.
6. Si `chercheSucre(`f`)` et `vide(`p_2`)`,
   alors f se déplace sur p_2.

1.2 Simulation

La simulation consiste essentiellement à initialiser la grille en plaçant les fourmis, le nid, le sucre et les phéromones de nid, puis à itérer un certain nombre de fois le déplacement des fourmis. Il faut aussi diminuer périodiquement l'intensité des phéromones de sucre pour tenir compte de leur évaporation.

Initialisation de la grille avec les phéromones de nid

Les phéromones de nid ont une intensité décrite par un réel compris entre 0 et 1. L'intensité vaut 1 sur le nid et diminue linéairement quand on s'éloigne du nid. Pour initialiser la grille avec les phéromones de nid, on commence par donner l'intensité 1 aux cases du nid et l'intensité 0 partout ailleurs. Ensuite, si la grille est constituée de t lignes et t colonnes, on itère t fois la règle suivante sur toutes les cases :

$$\text{Si } (p < 1) \text{ alors } p \leftarrow \max(m - \frac{1}{t}, 0)$$

avec p l'intensité des phéromones de nid sur la case considérée et m la valeur maximale des phéromones de nid sur les cases voisines.

Déplacement des fourmis

Pour chaque fourmi f sur une case p_1, on essaie d'appliquer les règles une par une, de la plus prioritaire à la moins prioritaire. Pour chaque règle, on cherche une case p_2 parmi les cases voisines de p_1 qui vérifient la condition d'application de la règle. Si l'on trouve une telle case, on applique la règle. Sinon, on passe à la règle suivante. Si aucune règle ne s'applique, la fourmi ne se déplace pas.

Remarque sur la règle 6 : si l'on parcourt le voisinage toujours dans le même sens pour trouver une case vide p_2 où déplacer la fourmi, on va beaucoup plus souvent sélectionner p_2 parmi les cases observées en premier. La simulation ne paraitra alors pas très naturelle, les fourmis ne se déplaçant pas vraiment aléatoirement. Par exemple, si l'on cherche une case voisine vide en commençant par regarder la case au dessus, alors les fourmis se dirigeront très souvent vers le haut. Lors de l'application de la règle 6, vous devrez donc choisir la case p_2 aléatoirement parmi les cases voisines vides.

Phéromones de sucre

L'intensité des phéromones de sucre est un entier compris entre 0 et 255, qui diminue de 5 à chaque itération de la simulation. Lorsque la fourmi pose des phéromones de sucre, elle pose directement 255.

2 Affichage

Dans un premier temps pour le test, on pourra faire un affichage texte avec, par exemple, `'f'` pour représenter une fourmi sans sucre, `'F'` une fourmi avec du sucre, `'n'` pour le nid et `'s'` pour du sucre. Le défaut de cette méthode est qu'il sera difficile de visualiser les différents niveau de phéromone.

On utilisera donc un affichage graphique, ce qui permet d'avoir des couleurs. On représente chaque case de la grille par un pixel coloré. La couleur dépend du contenu de la case :

- les fourmis en noir
- le sucre en orange
- le nid en bleu
- la phéromone de sucre en rouge
- la phéromone de nid en vert

Si une case contient plusieurs éléments, on affiche en priorité l'élément qui est le plus haut dans la liste ci-dessus. L'amélioration suivante devra être apportée dans un deuxième temps : les phéromones seront affichées avec un dégradé permettant de rendre compte de leur intensité. Ainsi, les cases seront vert vif près du nid et de plus en plus claires quand la phéromone de nid décroît. De même, les cases seront rouge vif quand une phéromone de sucre vient d'être déposée et de plus en plus claires quand celle-ci s'évapore.

Pour l'affichage graphique, deux solutions sont possibles (on ne demande pas d'implémenter les deux):

2.1 Affichage non interactif

Dans ce mode, on va générer une suite d'images au format "ppm" dont les noms seront de la forme f000, f001, f002 .... représentant chaque étape de la simulation. On va ensuite utiliser le logiciel *beneton movie* pour les rassembler en un fichier gif animé. Il devrait être installé au bâtiment 336 et est téléchargeable gratuitement à l'adresse `http://www.benetonsoftware.com/Beneton_Movie_GIF.php`.

Le format des fichiers ppm est donné en annexe, il contient essentiellement un tableau de couleurs des pixels.

2.2 Affichage interactif

On peut alternativement utiliser la librairie graphique `libmlv` vue au premier semestre lors du TP de la semaine 9. Vous trouverez les détails sur les sujets :

`http://nicolas.thiery.name/Enseignement/Info111/Semaine9/TD.pdf`
`http://nicolas.thiery.name/Enseignement/Info111/Semaine9/TP.pdf`

Dans ce cas, le fait de cliquer sur une case rajoutera du sucre sur la case ce qui permet de nourrir sa fourmilière quand elle a mangé tout son sucre.

3 Implémentation de la simulation

Afin de vous aider à réaliser le projet, nous vous donnons une analyse de la suite d'actions à mettre en œuvre pour implémenter la simulation.

Àprès l'écriture de chaque fonction, il faut vérifier soigneusement qu'elle fonctionne bien. Soit quand c'est possible avec une autre fonction de tests que l'on **conservera**,

soit par un affichage. Dans ce second cas, on essayera également de conserver dans une procédure le code qui permet de reproduire ce test, sans le lancer automatiquement.

1. Spécifier **un type abstrait fourmi**, **un type abstrait place** (pour les cases) et **un type abstrait grille** contenant toutes les informations nécessaires pour la simulation.

   Le nombre de fourmis étant fixe, on ne fera pas un type abstrait "ensemble de fourmis", il suffit d'utiliser un tableau de fourmis.

2. Décider des types concrets associés.
   Aide 1 : Une fourmi doit être associée à ses coordonnées dans la grille. Pour cela, utiliser le type concret `coord` défini dans le TP9.
   Aide 2 : Comme une case peut contenir ou non une fourmi, s'il y a une fourmi, on identifiera celle-ci dans la case, en notant son indice dans le tableau des fourmis et on notera '-1' s'il n'y a pas de fourmi.

3. **Initialiser une grille** 20×20 avec au centre un nid de 2×2, 12 fourmis autour et deux tas de sucre loin du nid. L'intensité des phéromones de nid sera pour l'instant nulle partout. Le schéma suivant est un exemple de situation initiale où f = Fourmi, N= Nid, S= Sucre

									S										
								f	f	f	f								
								f	N	N	f								
								f	N	N	f								
								f	f	f	f								
																		S	

4. Afficher le contenu de cette grille dans le terminal pour vérifier que l'initialisation a correctement été effectuée.
5. Écrire une procédure `deplacerFourmi` qui prend en paramètre une fourmi f, sa place p_1, une place p_2, et déplace f de p_1 vers p_2. Tester cette procédure en déplaçant une fourmi de la grille initiale.
6. Implémenter le prédicat `vide(p : place)`. À l'aide de la fonction `choisirCoordAleatoirement` du TP8, déplacer chaque fourmi vers une des cases voisines vides et afficher la grille à nouveau.
7. **Afficher la grille** initiale dans un fichier ppm et le visualiser avec "benetton movie".
8. Itérer 30 fois ce comportement et **générer un premier film** rassemblant les 30 images produites.
9. Initialiser la grille avec **l'intensité de phéromone de nid** associée à chaque case. Afficher l'intensité en phéromone de nid des cases de la grille dans le terminal pour vérifier que l'initialisation a été correctement effectuée.
10. Faire en sorte que dans les images générées, les cases vides soient affichées en dégradé de vert représentant l'intensité des phéromones de nid.

11. **Implémenter les prédicats** définis dans la section 1.1 comme des fonctions $\texttt{cond}_i$ retournant *vrai* si la condition de la règle i est vérifiée.
12. **Implémenter les règles 1, 2, 3 et 6** et déplacer les fourmis en suivant ces règles, sur 30 itérations. On doit observer quelques fourmis qui ramènent du sucre au nid.
13. **Rajouter les règles 4 et 5** permettant d'accélérer "exponentiellement" le rapatriement du sucre. Construire un film d'environ 300 itérations.
14. (optionnelle) A présent, lorsque les fourmis ramènent du sucre, le tas de sucre doit diminuer. Les fourmis peuvent vider successivement plusieurs tas de sucre, sans modifier les règles, du fait que le tas de sucre diminue lorsque les fourmis en chargent.

4 Organisation et évaluation du projet

Deux séances de TD et deux séances de TP porteront sur le projet. N'hésitez pas à poser des questions à vos encadrants de TP pour régler les problèmes que vous ne parvenez pas à surmonter. Il est indispensable de commencer à travailler sur le projet dès maintenant.

L'évaluation du projet se fera au travers d'une soutenance orale. L'organisation de la soutenance est la suivante :

- vous vous installez sur une machine, chargez votre programme et vérifiez rapidement que tout fonctionne comme la dernière fois que vous l'avez testé;
- les deux membres du binôme ont 10 mn pour présenter ensemble le résultat de leur travail (organisation du code, visualisation des résultats, discussion sur les problèmes rencontrés...);
- ensuite l'examinateur a 5 mn pour juger de votre maîtrise de la programmation. À partir de ce moment, vous serez considérés individuellement, et un effort important sera fait pour mettre en avant clairement votre implication dans le projet. Un étudiant ne peut prétendre être noté sur un code qu'il ne réussit pas à expliquer. Un étudiant absent à la soutenance aura zéro sur vingt.

Le principal critère d'évaluation est le nombre d'actions fonctionnant correctement. Nous prendrons également en compte :

- Qualité de présentation...
- Types abstraits : pertinence, choix alternatifs, utilisation dans le code...
- Qualité du code : modularité, lisibilité...
- Maîtrise du code ;
- Originalité : gestion de problèmes rencontrés, affichage d'informations intermédiaires...

5 Annexe

Un fichier contient une image décrite par la suite de caractères suivante :

1. Un "magic number" identifiant le type.
   Pour un fichier ppm il s'agit des deux caractères "P3".
2. Des espaces (blancs, TABs, CRs, LFs).
3. Une largeur L, formatée avec des caractères ASCII en décimal.
4. Des espaces.
5. Une hauteur H, également en ASCII en décimal.
6. Des espaces.
7. La valeur de couleur maximale, en ASCII en décimal. Comprise entre 1 et 65536.
8. Un retour à la ligne ou d'autres caractères espace.
9. Un tableau de H rangées, allant de bas en haut, chaque rangée contient L pixels de gauche à droite. Chaque pixel est un triplet des composantes RGB (c'est-à-dire rouge, vert, bleu) de la couleur, dans cet ordre. Tous les nombres doivent être précédés et suivis d'un espace. Les lignes ne doivent pas dépasser 70 caractères.

Voici l'exemple d'un fichier img000.ppm contenant une image de 16 pixels (4 par 4), représentant une diagonale rouge sur fond vert (les composantes RGB de la couleur rouge sont (255,0,0) et celles de la couleur verte sont (0,255,0)).

```
P3
4 4
255
255 0 0    0 255 0    0 255 0    0 255 0
0 255 0    255 0 0    0 255 0    0 255 0
0 255 0    0 255 0    255 0 0    0 255 0
0 255 0    0 255 0    0 255 0    255 0 0
```

Ce fichier ppm a été généré par le programme suivant :

```
#include <iostream>     // pour cout
#include <iomanip>      // pour setfill, setw
#include <sstream>      // pour ostringstream
#include <fstream>      // pour ofstream
#include <string>

using namespace std;
// variable globale permettant de creer des noms de fichiers differents
int compteurFichier = 0;
// action dessinant un damier
void dessinerDamier(){
  int i,j;
  int r,g,b;
  ostringstream filename;
  // creation d'un nouveau nom de fichier de la forme img347.ppm
  filename << "img" << setfill('0') << setw(3) << compteurFichier << ".ppm";
  compteurFichier++;
  cout << "Ecriture dans le fichier : " << filename.str() << endl;
  // ouverture du fichier
  ofstream fic(filename.str(), ios::out | ios::trunc);
  // ecriture de l'entete
  fic << "P3" << endl
      << 4 << " " << 4 << " " << endl
      << 255 << " " << endl;
  // ecriture des pixels
  for (i = 0; i < 4; i++){
      for (j = 0; j < 4; j++){
        // calcul de la couleur
        if (i == j) { r = 255; g = 0; b = 0; }
        else { r = 0; g = 255; b = 0; }
        // ecriture de la couleur dans le fichier
        fic << r << " " << g << " " << b << "    ";
      }
    // fin de ligne dans l'image
    fic << endl;
  }
  // fermeture du fichier
  fic.close();
}
int main (){
  dessinerDamier();
  return 0;
}
```