Algorithme de la mobilité

Deuxième année Informatique

Rapport de projet

Exploration par des fourmis

Développeurs :
Johan Chataigner
Laurent Genty

Responsable :
Arnaud Casteig

25 décembre 2019

Introduction

Dans ce projet, il est question de gérer une colonie de fourmi et d'assurer sa survie. Dans cette colonie, une reine va apparaître à un endroit de la carte et produire à intervalle plus ou moins régulier, des fourmis qui vont avoir comme tâche d'aller chercher de la nourriture et de la ramener à reine. Pour ce faire, elles devront creuser des tunnels sur les cases non explorées et déposer des phéromones sur leur chemin afin de pouvoir retrouver la reine et la nourriture. En effet, les fourmis n'ont connaissance que de leur environnement direct !

1 Approche algorithmique

1.1 Généralités conceptuelles

Afin de remplir notre objectif, nous avons créé plusieurs classes permettant de représenter les différents éléments d'une fourmilière : cellules de nourriture et obstacles, fourmis (reine, enfant), classe abstraite concernant le déplacement des fourmis (destinations). Afin de remplir ces points, nous avons effectué des changements concernant l'architecture de l'application et concernant les algorithmes.

1.2 Héritage de la classe WaypointNode

Dans la version de base, notre fourmi faisait des sauts entre les cellules de la carte afin de pouvoir se déplacer, mais aussi, n'héritait d'aucune autre classe particulière. En la faisant hériter d'une classe précédemment développée durant les TP, à savoir, `WaypointNode`, nous pouvions donc faire en sorte d'avoir les mêmes comportements que cette dernière : déplacement *smooth*, un système de gestion de destination déjà implémenté...

1.3 Changements concernant une cellule

Concernant la classe d'une cellule, nous avons fait plusieurs changements concernant le contenu de cette dernière : nous avons ajouté les notions des phéromones (indiquant un chemin vers de la nourriture et l'autre vers la reine), comprenant le temps de d'estompage mais aussi les intensités de ces dernières. Mais aussi, les éléments concernant le creusage d'une case (le temps nécessaire, si elle est creusée ou pas)...

1.4 Stratégie de la fourmi

Le plus intéressant dans ce projet est bien entendu le comportement de la fourmi : comment elle va interagir avec les éléments qui l'entourent ? Comment va t'elle traiter ces informations ? Quels *hook* seront utilisés afin de traiter les nouvelles informations captées ?

Toutes ces questions peuvent être répondues grâce à la Figure 1. Elle nous décrit le comportement d'une fourmi lorsqu'elle arrive à destination (on considère que quand elle naît, elle arrive à destination et commence son algorithme par cela).

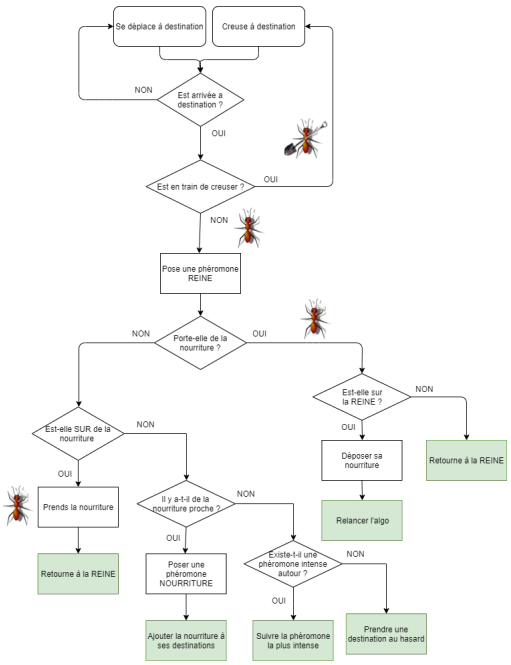

FIGURE 1 – Algorithme d'une fourmi à son arrivée à destination

Comme développé dans la classe Mère, la fourmi a des destinations et lorsqu'elle arrive à l'une d'entre elles, elle lance l'algorithme d'une fourmi. Une fourmi va donc vérifier si elle est arrivée à destination. Si tel est le cas, si elle creuse, elle doit finir de creuser la case (nous avons fait le choix que lorsqu'elle décide de creuser une case, elle s'arrête lorsque c'est fini, mais nous aurions pu très bien faire en sorte qu'elle décide d'arrêter à chaque `onClock()`).

Par la suite, si elle a fini de creuser sa case elle va s'y rendre, déposer une phéromone pour indiquer le chemin qu'elle a emprunté depuis la reine. Ensuite, si elle porte de la nourriture cela signifie qu'il faut qu'elle rentre déposer sa nourriture vers la reine (elle va essentiellement suivre les phéromones reine les plus intenses qui vont la mener vers cette dernière).

Si elle ne porte pas de nourriture, alors elle va sentir s'il y en a autour d'elle : si oui, elle va en destination de la nourriture tout en mettant une phéromone de nourriture (pour indiquer à ses amies qu'il y a de la nourriture dans ce coin) et va retourner vers la reine (en posant sur chacune de cellules qu'elle va emprunter pour rentrer une phéromone de nourriture).

Si au contraire il n'y a pas de nourriture autour, elle va prendre la phéromone de nourriture la plus intense autour si elle existe et s'y diriger dans l'espoir de trouver de la nourriture à proximité de cette dernière. Si une telle phéromone n'existe pas, alors elle va prendre une destination aléatoire.

La fourmi va donc avoir un comportement comme celui ci. Cependant, il faut ajouter à cela que nous avons fait le choix de creuser des cases en un certain nombre de rounds, ie. une certain nombre de `onClock()` : le coût d'une case (plus une case est profonde, plus son coût est élevé, plus le temps de creusage prendra de rounds).

A chaque tour d'horloge, si une fourmi est en train de creuser, on va creuser un tour de plus, réduire la "couleur" de la case (lorsqu'une case est creusée entièrement, elle apparaît blanche) et si la case a enfin finie d'être creusée (seulement grâce à cette fourmi ou à plusieurs) alors enfin elle pourra s'y diriger. Attention cependant, les fourmis ont une espérance de vie moyenne entre 500 et 1000 tours d'horloge, ce qui fait qu'il faut vite les creuser !

Enfin, le comportement pour aller à une cellule est un peu différent que celui tout simple de notre classe mère, c'est donc pourquoi nous avons fait le choix de surcharger la méthode. En effet, si l'on souhaite ajouter une destination, il faut prendre en compte les cases sur lesquelles nous venons tout juste de passer. En effet, malgré le fait qu'une fourmi n'a connaissance que de son environnement direct, nous avons fait le choix d'ajouter les 2 cases précédentes par lesquelles nous sommes passés afin d'éviter le phénomène de *Death Spiral*, évènement létal pour nos petites fourmis. Le comportement supplémentaire dans cette méthode est de vérifier lorsqu'on ajoute une destination la case que l'on souhaite atteindre est creusée, si oui on y va (comportement "normal" de la classe mère) sinon on se met dans un état de creusage et donc le comportement précédemment énoncé s'effectuera.

Concrètement, une fourmi va déposer des phéromones à l'aller afin d'indiquer d'où est-ce qu'elle vient pour trouver la reine, et lorsqu'elle aura trouver de la nourriture, elle deposera des phéromones indiquant son chemin vers de la nourriture.

2 Difficultés et problèmes rencontrés

Au cours de notre développement nous avons rencontré plusieurs problèmes. Malgré l'aspect simpliste du projet, il s'avère que nous avons eu plusieurs soucis concernant les phéromones. En effet, dans la version actuelle du projet, nos fourmis sont parfois induises en erreurs par elles mêmes dans la mesure où les phéromones ne les amène pas dans la bonne direction. Ce qui fait que le phénomène de *Death Spiral* est très présent. On se retrouve parfois dans des situations avec des dizaines de fourmis effectuant des triangles (ou cercles) en train de desesperement chercher leur chemin. Ce phénomène induisant les fourmis en erreur a d'autres conséquence. En effet, cela veut dire que même si une fourmi a trouver de la nourriture et qu'elle l'a ramène à la reine, quand elle voudra repartir, elle ira dans la mauvaise direction et ne retrouvera pas la cellule de nourriture qu'elle avait trouvée.

Le deuxième problème majeur est le fait de la dissociation entre le creusage (se faisant dans le `onClock()`) et le déplacement. En effet, si une fourmi souhaite creuser une case, elle le fera en X tours d'horloge, mais il se peut que le déplacement de la fourmi se fasse en plus de temps : ce qui veut dire que lorsqu'une fourmi aura fini de creuser une case, elle pourra en creuser une autre (cela se voit sur un Run de l'application) pendant qu'elle sera toujours en train de marcher pour atteindre la case (ce problème est partiellement résolu en augmentant la vitesse d'une fourmi).

3 Objectifs atteints et améliorations possibles

Nous avons rempli plusieurs objectifs : faire en sorte que les fourmis prennent de la nourriture et la ramène à la reine (lorsque les phéromones et le *Death Spiral* nous le permettent). Les fourmis peuvent aussi bouger sur la carte en creuser une case si ce n'est pas déjà fait. Elles peuvent poser des phéromones afin de se diriger (et les phéromones ont une intensité maximale et une durée de vie maximale aussi). La durée de vie limitée des fourmis, de la nourriture et de la reine (si elle épuise ses ressources) a été développé et est fonctionnel, le creusage d'une case selon sa profondeur (et donc selon son coût) a aussi été fait. Enfin, l'apparition de la nourriture selon la profondeur et les obstacles (qui ne peuvent être franchis par la nourriture ou les fourmis) ont aussi été implémentés.

Les prochains axes d'améliorations après la correction des problèmes cités seraient de faire en sorte que les fourmis s'échangent des messages concernant la nourriture (la quantité de nourriture qu'il y avait quand une fourmi a pris de la nourriture) et donc potentiellement permettre d'optimiser les déplacements et la pose des phéromones.