Adrian MOREL,
Luca JIMÉNEZ GLUR

Rapport du rendu 3 : projet TCHANZ

23.05.2022
POO, BA2

1. Description de notre approche

Fourmi generator : la fourmi generator tente de se déplacer vers le coin inférieur gauche de sa fourmilière. Ceci lui donne un avantage face aux rétrécissements possibles de la fourmilière, car ils se font toujours en conservant la position du coin inférieur gauche. Sinon, des fourmis situées en bas à gauche pourraient empêcher la generator de rester dans la fourmilière lors de rétrécissements de cette dernière.

Naissance de fourmis : les fourmis naissent au premier emplacement libre trouvé dans la fourmilière en la balayant à partir du coin supérieur droit de manière verticale, pour que le spawn se fasse loin de la generator. Ceci fait que les defensor sont plutôt près du coin supérieur droit de la fourmilière, donc quand elle rétrécit les defensors meurent assez facilement.

Fourmi collector : en cas de 2 chemins équivalents, celui qui est choisi est celui qui va vers l'ouest si la nourriture est au nord ou au sud, et celui qui va vers le nord si la nourriture est à l'est ou à l'ouest (choix arbitraire). Une fourmi collector sans nourriture cible attend qu'une nourriture atteignable apparaisse sans rien faire, mais si elle est dans sa fourmilière elle tente de sortir de cette dernière en respectant les priorités de la donnée.

Fourmi defensor : le but de la fourmi defensor est de rester à la frontière de sa fourmilière. Pour cela, la defensor trouve d'abord le coin de la fourmilière le plus proche de son corps, puis le coin le plus proche parmi les deux coins adjacents et elle se dirige vers le bord formé par les deux coins retrouvés. Si la fourmilière rétrécit, la defensor se superpose au bord, elle cherche donc la direction qui lui permet de rester à l'intérieur de la fourmilière.

Fourmi predator : lorsqu'une fourmi predator n'a pas de fourmi cible et que sa fourmilière est dans l'état CONSTRAINED, elle va simplement attendre. Dans le cas où sa fourmilière serait dans l'état FREE, la fourmi predator va chercher à rentrer à l'intérieur de sa fourmilière si elle ne l'est pas déjà. Ensuite, elle va patienter jusqu'à ce qu'une fourmi collector ou predator d'une autre fourmilière entre dans son territoire. De façon générale, la fourmi predator considère un déplacement seulement vers les deux positions qui se trouvent dans le même quadrant que la cible. La position choisie au final est celle qui la rapproche le plus de la cible. Si sa fourmilière est envahie, la fourmi predator rentre chez elle.

2. Captures d'écran

Fichier test fourni (f10.txt):

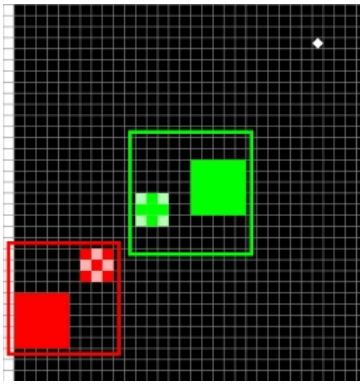

Figure 1 (image de gauche) : ce fichier test est composé de deux fourmilières et d'une seule nourriture. La fourmilière verte a une fourmi defensor qui est positionnée vers la bordure de cette dernière pour la protéger. La fourmilière rouge possède une fourmi collector qui se dirige vers l'unique nourriture présente dans la carte. Sa bordure est superposée au bord du monde, ce qui est autorisé.

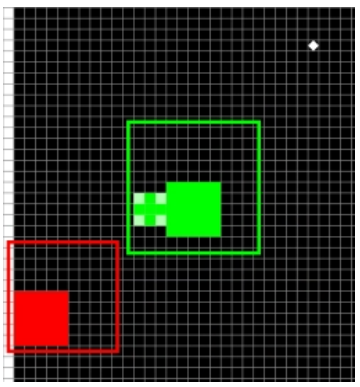

Figure 2 (image de droite) : en suivant le chemin le plus court pour atteindre la nourriture, la fourmi collector a rencontré la fourmi defensor verte. Au contact, la fourmi rouge est tuée. On peut aussi remarquer que la fourmi generator verte s'est déplacée pour aller en direction du coin inférieur gauche.

Fichier test choisi :

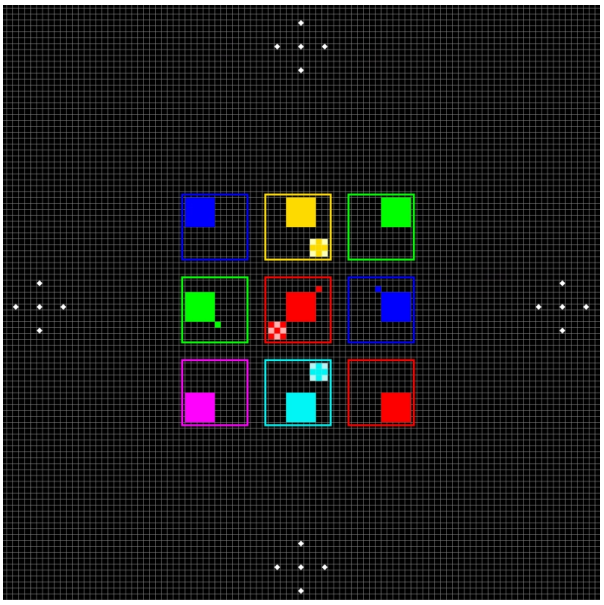

Figure 3 (gauche) : initialement, il y a 9 fourmilières en formation de carré entourées de 4 îlots de nourriture. Les generators ont des positions arbitraires et les fourmilières désavantagées (car elles risquent d'être CONSTRAINED) ont quelques fourmis par défaut.

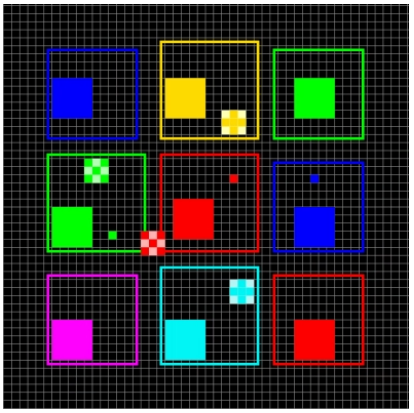

Figure 4 (droite) : après 3 étapes, la generator de chaque fourmilière se dirige vers le coin inférieur gauche de sa fourmilière. La collector de la fourmilière centrale part à la recherche de nourriture, mais entre dans le territoire de la fourmilière verte à gauche, ce qui active sa predator. Une collector verte née à l'étape 2 part aussi vers une nourriture. Les fourmilières qui avaient des fourmis par défaut se sont agrandies d'une unité.

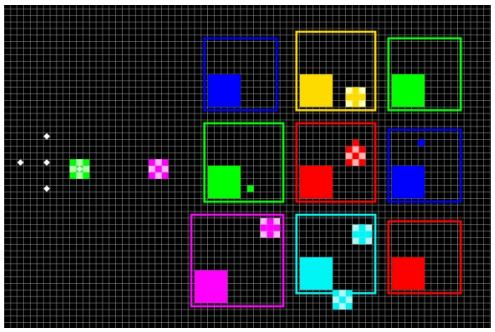

Figure 5 (gauche) : après 28 étapes, la collector verte capture une nourriture. Une collector rouge, une magenta et une cyan ainsi qu'une defensor magenta sont nées entre-temps. Les defensors restent collées au bord. La fourmi collector rouge est bloquée depuis plusieurs step contre la fourmi predator.

3. <u>Méthodologie et conclusion</u>

Chaque semaine nous avons fait entre une et trois séances de programmation à deux en présentiel : nous discutions de méthodes de résolution de problèmes, et en alternance une personne codait et l'autre surveillait et conseillait. Nous avons trouvé cette méthode efficace pour avancer dans des tâches complexes. À partir du rendu 3, nous avons fait des séances supplémentaires suivant la même dynamique avec Discord. Environ ¾ du travail a été effectué en simultané et le reste de manière individuelle. Nous avons trouvé cette proportion adéquate, car elle permet d'éviter les situations où il faut rassembler beaucoup de codes différents et il est aussi plus facile de se retrouver dans le code et de bien le comprendre, tout en évitant de la potentielle duplication de fonctions.

Le bug qui a posé le plus de problème était par rapport à la grille de booléens dans squarecell. Pour la lecture de fichier, nous avons indiqué dans la fonction de dessin dans cette grille que si une erreur avait été détectée, il fallait ne rien faire. Mais il se trouve que des erreurs pouvaient être détectées lors d'une mise à jour de la simulation avec des carrés créés hors du monde. Ceci a fait que les entités qui avaient bougé après la détection d'erreur n'étaient pas redessinées dans la grille de booléens et pouvaient donc se superposer dans le monde. Le problème a été résolu en ajoutant un booléen static dans squarecell.cc qui indique si une lecture est en cours ou pas. Si le booléen est faux, on ignore la détection d'erreurs et on peut dessiner normalement dans la grille de booléens.

L'environnement de travail proposé par Geany est convenable pour les premiers projets. Un point positif du projet est son aspect ludique, il est amusant de créer une simulation représentant l'évolution d'une fourmilière, face à la concurrence et au besoin de nourriture. Comme point négatif, nous pouvons noter la lourdeur de la syntaxe de GTKmm. En effet, il est assez difficile de manipuler l'interface graphique sans devoir toujours se référer à internet ou à du code précédent. Le projet nous a permis de progresser fortement en implémentant les notions de programmation orientée objet.