## a) Description de notre approche :

Pour les déplacements non-précisés de la donnée générale, nous avons choisi de procéder comme suit :

- **Generator :**

⇨ Conserve-t-elle sa position initiale ?

La fourmi generator se déplace de 1 pas vers le centre (centré) de la fourmilière après avoir analysé les positions autour de la fourmilière. Elle se déplace donc en fonction de ce qui entoure sa fourmilière et toujours vers le centre.

- **Lieu de naissance des fourmis :**

⇨ Comment est-il choisi ?

Nous avons choisi d'établir le lieu de naissances des fourmis par ordre ascendant sur l'axe des abscisses, et par ordre descendant sur l'axe des ordonnées : en commençant par le côté haut gauche, vers la droite, puis en descendant.

- **Collector :**

⇨ Quel chemin est choisi en cas de 2 chemins équivalents ?

En cas de deux chemins équivalents, la fourmi collector a un chemin de préférence en fonction de la position de la nourriture vers laquelle elle souhaite aller.

En effet, deux compteur, initialisés à 0, comptent le nombre de superpositions pour chaque chemin possible. La fourmi collector choisi donc son chemin en fonction de ces compteurs. Par exemple, si le premier compteur est inférieur ou égal au deuxième, la fourmi collector suivra le premier chemin. Or si le deuxième compteur est strictement inférieur au premier, elle suivra le deuxième. La fourmi collector a donc à chaque fois un chemin de « préférence », qui se met en place si les compteurs sont égaux.

⇨ Que fait une fourmi Collector sans nourriture cible ?

Une fourmi collector sans nourriture cible reste statique.

- **Defensor :**

⇨ Comment son but est-il mis à jour ?

A sa naissance, une fourmi defensor se déplace vers la bordure de la fourmilière la plus proche de sa position. Lorsqu'elle se superpose avec une bordure, son booléen « end_of_life » passe à true et elle meurt. Une fois sur sa bordure, elle tue chaque collector qui rentre en contact avec elle.

- **Predator :**

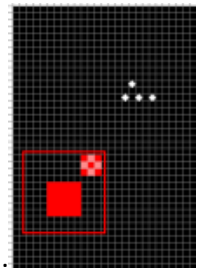
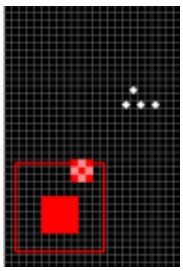
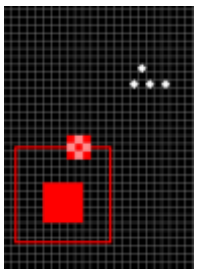
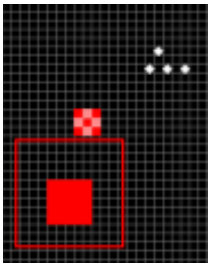
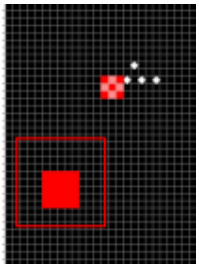
⇨ Que fait une fourmi predator sans fourmis collector cible ?

Une fourmi predator sans fourmi collector cible se déplace aléatoirement tant qu'elle ne superpose pas avec quelque chose déjà présent, et qu'elle ne sort pas de la grille.

## b) Captures d'écran de plusieurs étapes de notre simulation :

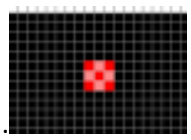
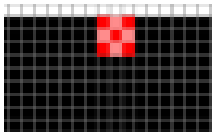
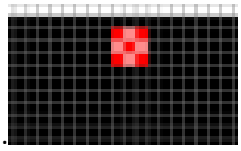
### a. Fichier test fourni : Test f08.txt

**Départ** **Collector reste bloquée à l'infini**

1.  2.  3.  ..... 4.  5. 

Le collector va tout d'abord regarder quel est le chemin présentant le moins de superposition. Il décide donc de prendre le chemin du haut et va se déplacer en plusieurs steps en haut à gauche jusqu'à se retrouver sur la même diagonale que sa cible. Dès qu'elle est sûr la même diagonale que sa cible elle suivra ce chemin jusqu'à

### b. Fichier de notre choix : Test personnel composition suivante : T118.txt

1.  2.  3. 

Le collector rebondit contre la paroi afin de quand même suivre le chemin le plus optimal tout en restant dans le cadre.

1.  2. 

Le collector rouge entre en collision avec le defensor vert (d'une autre fourmiliere), et meurt instantanement en laissant sa nourriture sur place. Le collector s'etant déplacé le temps d'une simulation, la nourriture s'en retrouve a la position actuelle en 2eme step.

## c) Méthodologie et conclusion :

En tant que groupe nouvellement reconstitué composé de 3 personnes, nous avons choisi pour ce troisième rendu, d'organiser notre travail de la manière suivante :

- **EL QABLI Rim :**
  - Modules Fourmi, Fourmilière et Simulation :
    Naissance et mort (fourmis collectors/predators au contact de fourmis predators d'autres fourmilières, et fourmis collectors au contact de fourmis defensors d'autres fourmilières)
- **FUMEAUX Max :**
  - Module Fourmilière : Changement de taille, mouvement de generator
  - Module Nourriture : apparition, consommation de nourriture par la fourmilière
- **SCHWARTZ Maxime :**
  - Module Fourmi : Déplacement des fourmis

Afin d'organiser notre rendu au mieux, nous avons procédé par une vérification et débuggage individuel de chacune des parties de chaque membre du groupe, pour ensuite mettre toutes les parties en commun. Cela a permis de considérablement réduire les bugs potentiels. Nous avons donc pu consacrer 60% de travail en mode individuel, pour 40% du travail en groupe afin de tester l'intégralité de notre programme. Nous trouvons que cette répartition reste la plus optimale, et pour cette raison, nous ne modifierons pas cette proportion.
Tout au long de ce rendu, nous n'avons pas rencontré un seul bug en particulier. Quant à celui qui nous a posé le plus de problèmes, il s'agit du fameux *segmentation fault*. Nous avons pu résoudre ce problème grâce à une minutieuse vérification des indices de vectors, afin de vérifier si l'on n'appelle pas un élément inexistant du vector, ce qui crée un problème de mémoire.
En définitive, nous pouvons a posteriori émettre en tant qu'auto-évaluation de notre travail la suivante. Nous estimons que les points forts de notre rapport restent l'écoute, la communication, entre trinômes et l'efficacité dans le travail. de projet. Si l'on devait soulever un point faible que l'on pourrait éventuellement améliorer, ce serait la gestion du temps afin de pouvoir rendre le projet quelques jours à l'avance.