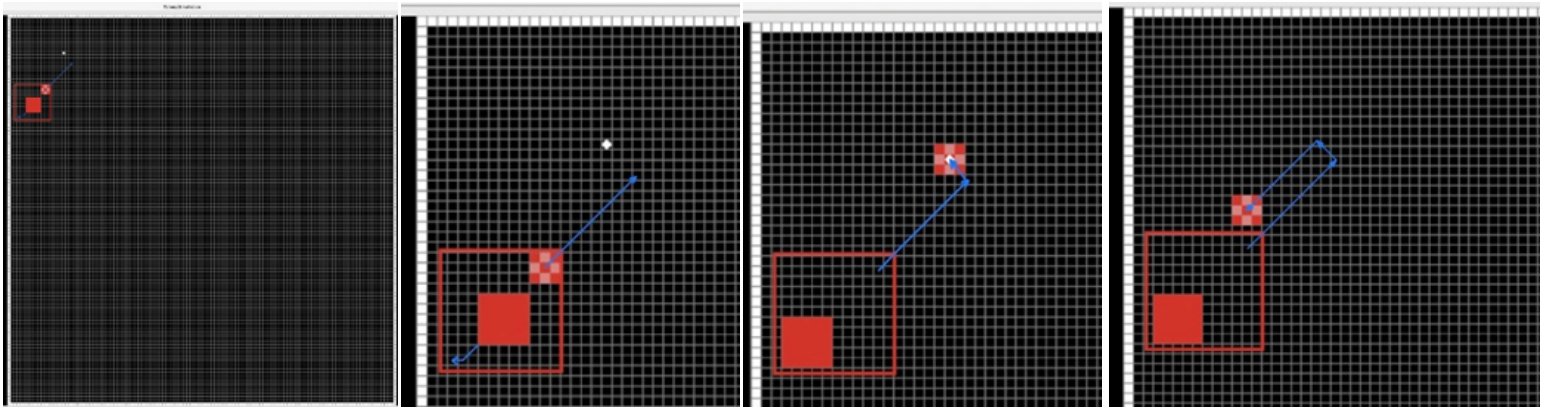
Emilien Coudurier sciper : 345431
Aksel Acar sciper : 344405

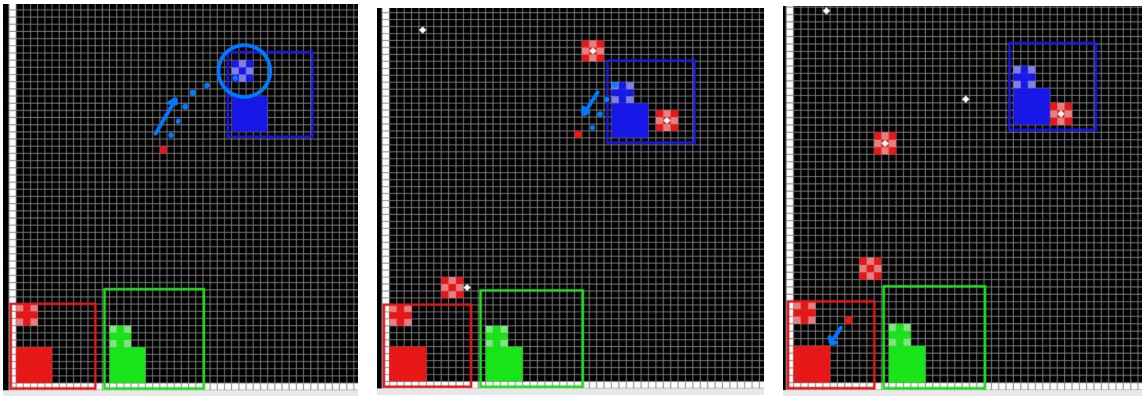
Generator : Les fourmis Generator ne conservent pas leur position initiale. Nous avons décidé de les faire se déplacer vers le coin inférieur gauche de leur fourmilière, afin de permettre aux fourmis de naître même si la taille de la fourmilière est petite. Ce déplacement permet de fluidifier les mouvements des autres fourmis au sein de leur fourmilière. Nous avons choisi le coin inférieur gauche étant donné qu'il s'agit du dernier coin par lequel la fourmilière peut s'agrandir. Ainsi, ceci limite les déplacements de la Generator.
Naissance : Le lieu de naissance est choisi par un algorithme qui balaye la fourmilière de haut en bas et de la gauche vers la droite. Dès qu'un espace libre sans superposition est détecté la fourmi est posée. De cette manière, la majorité des fourmis naissent au bord de la limite supérieure et ainsi les Defensor n'ont pas beaucoup de mouvements à effectuer étant donné qu'elles sont déjà sur le bord.
Collector :

- Dans le cas où 2 chemins sont équivalents les fourmis Collector vont regarder le chemin ayant le moins de superposition et emprunter celui-ci. Si les chemins sont 100 % identiques, elles vont prendre celui par rapport à la projection de la droite ayant pour vecteur directeur (1, 1).
- Une Collector sans but rentre à sa fourmilière en empruntant le chemin le plus court.

Defensor : Le but de la fourmi Defensor est de rester sur les bordures de sa fourmilière et de s'adapter aux variations de taille dûes à sizeF. Son but premier est d'aller contre le côté de sa fourmilière le plus proche. Si celui-ci est accompli, elle se dirigera vers le coin le plus proche. Enfin la fourmi Defensor s'adapte aux variations de taille de la fourmilière pour ne pas être en superposition avec celle-ci et donc pouvoir survivre. Elle mourra cependant si elle se trouve dans l'angle qui réduit de taille.
Predator : Une Predator sans but rentre à sa fourmilière en empruntant le chemin le plus court, restant autour de sa fourmi Generator.



Etat initial : La Collector va se déplacer en diagonale pour chercher la nourriture. La Generator va se déplacer vers le coin inférieur gauche.
Après 11 mises à jour : La collector a atteint la nourriture. Son booleén has_food passe à vrai. La Generator ne bouge pas étant donné que la coin n'a pas bougé .
Après 18 mises à jour : La Collector prend le chemin le plus court vers sa Generator. Dès qu'elle est en contact avec sa fourmilière elle dépose la nourriture et total_food de la fourmilière augmente.



Emilien Coudurier sciper : 345431
Aksel Acar sciper : 344405

Etat Initial : La predator de la première fourmilière se dirige vers le collector le plus proche en dehors de sa fourmilière, puisqu'elle est dans l'état constrained. Les collectors vont chercher les nourritures pour les ramener à leur fourmilière et les defensors se dirigent vers les coins (ici contre la Generator).
Après 12 mises à jour : Plus aucune collector n'existe dans le monde à part celle de la première fourmilière, la predator cherche donc à revenir à sa Generator.
Après 27 mises à jour : La predator est rentrée dans sa fourmilière, mais continue jusqu'à sa Generator. Les Collectors continuent leurs mouvements pour récupérer les nourritures.

Pour répartir le travail et nous organiser au mieux nous avons pris un certain temps au début de chaque rendu pour lire la donnée et découper la tâche en différents éléments principaux Ceci nous a permis de bien répartir les tâches et donc d'être efficace lors du travail individuel. Nous nous consultons régulièrement afin d'être à jour sur l'avancement de chacun et de pouvoir nous entraider en cas de bug/difficultés. Même si chacun travaille sur une partie du projet, nous étions souvent soit l'un à coté de l'autre soit en appel afin de pouvoir communiquer et d'être sûrs que ce que l'un faisait ne contraigne pas l'autre et que les solutions envisagées étaient les mêmes dans leur principe.

	Emilien Coudurier	Aksel Acar
graphic	60%	40%
squarecell	10%	90%
nourriture	50%	50%
fourmi	75%	25%
fourmilière	50%	50%
simulation	90%	10%
gui	70%	30%

Nous avons commencé par les modules squarecell et simulation étant donné que ce sont les piliers du projet. En effet, le bon fonctionnement des ces 2 modules est primordial pour tous les autres modules. Nous avons passé beaucoup de temps à tester le module squarecell et vérifier que tous les cas limites marchent bien. Pour effectuer ces tests nous avons cout une autre grille de booléens dans le terminal pour voir où sont posés les carrés. De plus, lors des tests, nous avons réduit la taille de notre grille afin que ça soit plus facile d'identifier des erreurs de placement des carrées.
Quant aux tests du module simulation, ils consistaient à la création petit à petit d'un fichier texte, en testant la bonne lecture de chaque élément rajouté, au fur et à mesure. Des tests de fichiers incorrects déterminés grâce à la grille de squarecell ont aussi permis d'aboutir à un module complet et fonctionnel.

La proportion de travail simultané est d'environ **35%** pour **65%** de travail indépendant. Avec du recul nous ne changerions pas ces proportions. On a trouvé qu'en répartissant ainsi le travail nous arrivions à être efficaces. Diminuer le temps de travail simultané aurait pu être bénéfique à court terme mais cela aurait pu nous poser des soucis lors de la mise en commun. Nous aurions donc perdu du temps à réarranger le code pour qu'il puisse être assemblé et fonctionner correctement ensemble.
Le bug le plus fréquent auquel nous avons fait face est celui du segmentation fault dû à des mauvaises incrémentation de boucles for qui faisait que le code essayait d'effectuer des opérations pour des parties de vector inexistantes. Un autre souci qui causait des segmentations fault était celui des pointeurs sur les Fourmis. Des espaces mémoires mal désalloués et des pointeurs pointant sur des entités inexistantes causaient ces problèmes.

Nous avons su travailler en binôme de manière efficace et méthodique, ce qui nous a permis de finir tous les rendus à temps. Les assistants ont été d'une aide précieuse et ont été très pédagogiques. Cependant, les données étaient parfois imprécises alors que la correction ne laissait que peu de place à une liberté dans l'écriture du code. Des indications plus précises concernant les libertés qui peuvent être prises seraient intéressantes. Les rendus étaient équilibrés en terme de temps de travail, ce qui évitait les mauvaises surprise compte tenu du temps qui nous était donné.