2.5.1 Décision de création d'un nouveau robot

Nous donnons la priorité à la création des réparateurs, donc s'il y a un neutraliseur qui est en panne et qu'il n'y pas de réparateur déjà sur le terrain, un réparateur visant le neutraliseur le plus proche sera créé s'il arrive à atteindre le robot à temps.
Un neutraliseur sera seulement créé s'il n'y a pas besoin d'un réparateur et s'il y a moins que 3 neutraliseurs en service. Il sera aligné avec la particule la plus grande sur le terrain. Vu que le réparateur n'a pas besoin de s'aligner, on ne fait que de le créer et une autre fonction lui assigne sa cible.
Nous contrôlons pour les deux cas s'il n'y a pas de superposition avec un robot en service lors de la création du robot.

2.5.2 Décision de mise à jour des robots réparateurs

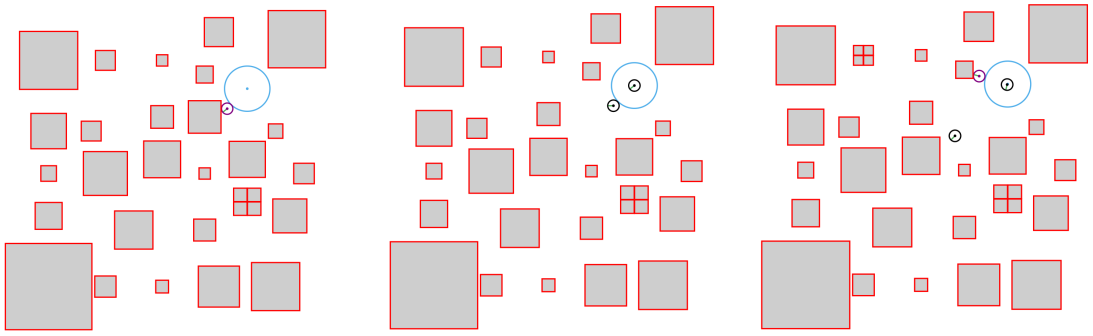
Les réparateurs sont simplement assignés au neutraliseur le plus proche.

2.5.3 Décision de mise à jour de la particule cible des robots neutraliseurs

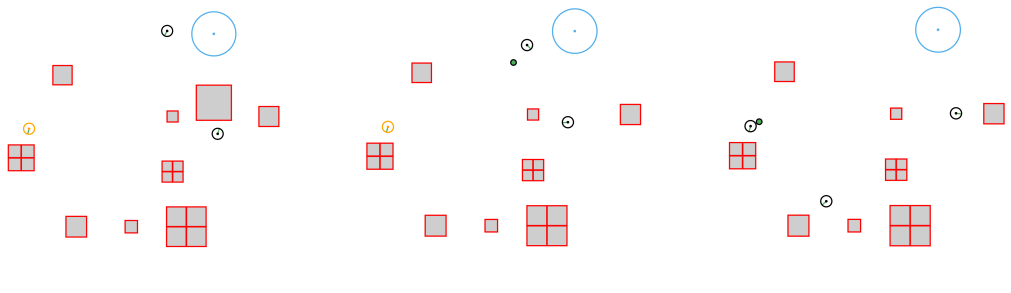
Les neutraliseurs sont assignés à la particule la plus grosse, et la plus proche. Les particules étant triées par ordre de taille décroissants, on a juste à itérer sur cette liste. Dans ces itérations, après avoir trouvé le neutraliseur le plus proche de la particule, on contrôle si celui-ci ne serait pas plus proche d'une autre particule de même taille, et si c'est le cas, on l'y envoie. Une particule ne reçoit donc qu'un neutraliseur au maximum.

Captures d'écran

Ici, le robot neutraliseur a été assigné pour détruire la particule la plus grosse, c'est-à-dire celle en bas à gauche. Il rencontre déjà un obstacle sur le chemin. Il va donc pivoter pour s'aligner avec lui et le détruire. Il reprend ensuite sa route vers sa cible.



Dans la situation ci-dessous, le robot est en panne en 1671. Un robot de réparation va donc être émis par spatial en 1700. Le robot sera réparé en 1989.



Méthodologie et conclusion

Nous avons utilisé Clion pour le code et Git sur GitHub pour permettre la collaboration. Le but était d'avoir une répartition de travail équitable, donc on a travaillé en collaboration au sein des différents modules. On a commencé par le module le plus bas et quand on le finissait, on le testait, ainsi on pouvait faire confiance aux modules de bas niveau et on évitait de devoir descendre toute la hiérarchie quand on avait des erreurs plus tard. Pour tester le module, on imaginait les cas limites avec un ou deux qui étaient généraux et on mettait le module à l'épreuve. Nous avons fait environ 75% du travail côte à côte, ce qui permet de bien être d'accord l'un avec l'autre et de suivre la progression de chacun. Nous trouvons cette proportion correcte, car elle permet aussi de travailler tout seul sur les fonctions qui sont longues à coder ou encore de corriger les erreurs que nous avons remarquées côte à côte.

Le bug le plus fréquent était en lien avec le passage par référence. Nos getters renvoyaient simplement la valeur, et on la changeait mais l'attribut ne se modifiait pas. On a donc simplement renvoyé tous nos getters par référence.

Le bug qui nous a posé le plus de problèmes était lors de l'étape ou il fallait associer une particule comme cible pour chacun des robots. Nous avons réussi à le réparer en créant un attribut dans robot (job_) et dans particule (cible_) qui empêchent un double assignment évitant ainsi des erreurs.

En conclusion, notre programme permet à un robot de changer de cible à mi-chemin, s'il y a un autre qui est plus proche. Un neutraliseur recevra toujours comme cible la plus grande particule sur le terrain. Un réparateur se dirige vers spatial s'il n'y a pas de neutraliseur à réparer, ça évite des collisions imprévues.

Nous voyons des améliorations à faire dans la création d'un nouveau robot neutraliseur. On pourrait "dire" à spatial qu'il y aura un robot dans la prochaine mise-à-jour, passer par la fonction qui assigne les cibles et ensuite créer le robot déjà aligné avec sa cible. De plus, on aimerait améliorer les cas de collisions entre robots neutraliseurs, peut-être les faire aligner avec leur cible même s'ils ne la touchent pas, car ça permettrait un déblocage naturel.