Sara Conti et Elio Wanner - 21 mai 2021

Rapport - Rendu 3

Description des algorithmes:

Algorithme de connection:

Notre stratégie de connexion a lieu en deux étapes. La méthode updateConnections a lieu en deux étapes:

- La première étape est récursive. Elle consiste à déterminer les robots de communication du centre des bases et de trouver les robots qui y sont connectés. Cela remplit l'attribut du vecteur connectedRobots du robot du centre de la base.
- La deuxième étape consiste à prendre chaque robot d'une base et de voir si ce robot est dans la liste de connectedRobots du robot du centre de la base. Si c'est le cas, le booléen remote de ce robot passe à true. Cette étape n'est pas récursive et nécessite donc moins de ressource, nous évitant de faire la première étape pour tous les robots.

Si on exécutait la première étape pour tous les robots, cet algorithme aurait été le plus coûteux de notre implémentation. La méthode UpdateNb qui est utilisée pour la première étape prend probablement plus de ressources que la méthode updateConnections, car elle compare tous les robots entre eux.

Stratégie concernant les robots:

Notre stratégie contient différentes étapes de déploiement des robots.

La première étape a lieu uniquement si la base ne possède pas déjà à priori plus de 3 robots de prospections. Elle consiste à faire partir dans 4 directions orthogonales (N,E,S,O) 2 robots de communications et un robot de prospection. Les deux robots des communications permettent au robot de prospection de parcourir une plus grande distance tout en restant connecté à la base. Une fois que les robots de prospection ont atteint leur but, ils retournent à la base pour se recharger si besoin ou pour reprendre contact avec elle, afin d'avoir un nouveau but. Ce but est ensuite redéfini par la base à une position aléatoire donnée, pour ensuite repartir dans une direction à celle précédente (N-E). Cela nous permet de baliser une plus grande surface de la fenêtre car tous les robots n'ont pas comme point de départ de leur trajectoire la base en elle-même. Chaque robot aura la même direction aléatoire à chaque lancement du programme, afin que le résultat de la simulation soit toujours le même.. Il en va de même pour les robots préexistants de la base.

Ensuite arrive l'étape de la création du réseau de robots de communication. Elle a lieu lorsque la base a au moins 1% de ses ressources finales nécessaires.

Enfin, lorsque la base atteint son état d'autonomie, les robots de communication de cette base restent dans leurs positionnement en réseau.

A chaque fois qu'un robot de prospection découvre un gisement, sa base produit un robot de forage et de transport qui auront pour but la position du gisement trouvé. Le robot de forage atteint son but ultime, il ne va plus bouger de la surface du gisement. Le robot de transport va faire des allers-retours entre le gisement et la base pour ensuite terminer son parcours sur le centre du gisement.

Concernant la maintenance des robots, nous avons décidé de systématiquement maintenir les robots de prospection et de transport. Cela permet aux robots de prospection de continuer à chercher des gisements et aux robots de transport de continuer à transporter ces gisements. Ce n'est pas le cas des robots de communication et de forage qui ne retournent pas à la base après l'avoir quittée.

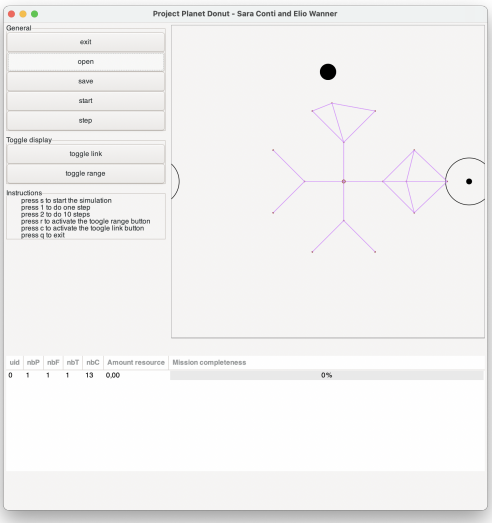
Robustesse de l'approche:
Normalement, lorsque le robot de prospection atteint son but, la base va changer son but toujours dans une direction donnée dans settings.h (en haut à droite). Afin d'éviter que tous les robots d'une base partent du même point et dans la même direction, les robots de prospection ont après chaque maintient un nouveau but aléatoire à partir duquel ils vont commencer leur trajectoire.

Méthodologie et conclusion:
Communs: Projet, makefile
Elio: modules utilities, simulation, robot, bases, gisement, graphic, settings.
Sara: modules geomod, gui + rédaction des rapports.
Nous avons souvent travaillé ensemble sur le campus pour que nos modules soient compatibles. Au début de chaque rendu, nous avons fait des mind-maps ensemble pour visualiser comment les différents modules communiquaient entre eux et où il était plus judicieux de placer chaque méthode.

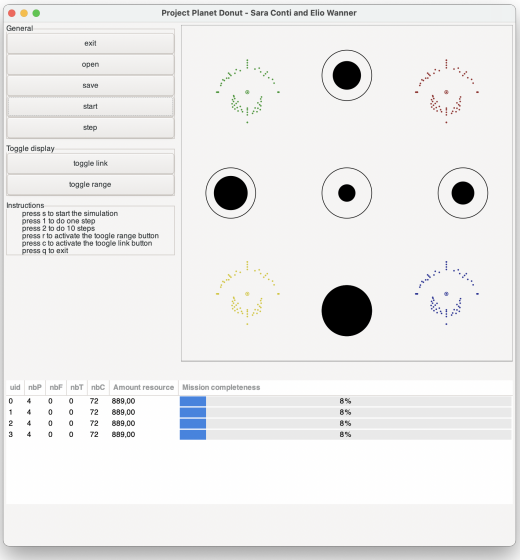
Erreur la plus fréquente:
invalid operands to binary expression ('point2D' and 'point2D') : difficulté à comparer des points 2D. Nous avons finalement utilisé la fonction isEqual.

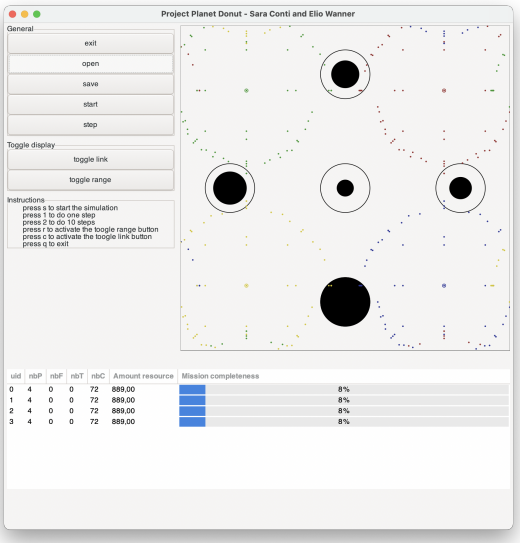
Auto-évaluation:
Nous sommes satisfaits du résultat obtenu. Nous pensons avoir bien géré notre temps en travaillant avec régularité. Cependant, malgré le fait que nous ayons investi une durée de temps similaire dans le projet, nous avons constaté que la rapidité de code était très fortement influencée par les connaissances de bases de chacuns.
Nous avons apprécié que vous preniez le temps de nous répondre personnellement sur discourse et que votre réponse était toujours très rapide, nous permettant d'avancer dans notre rédaction de code.

Fichier fourni (rendu3_image.txt):

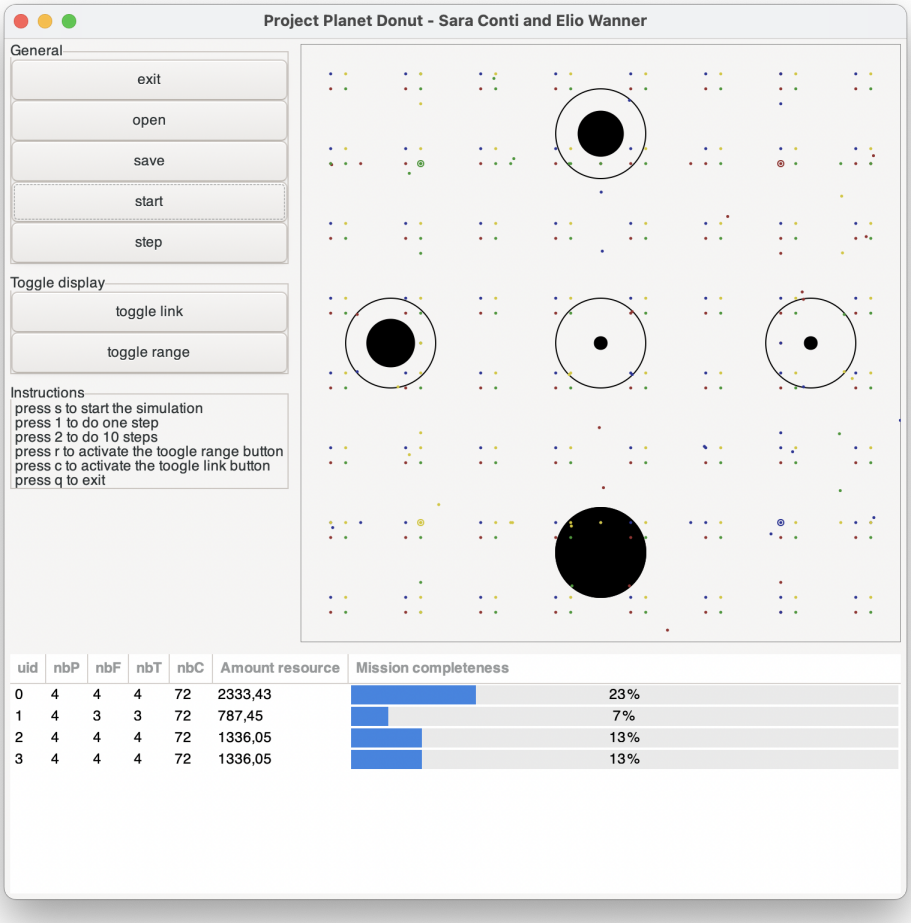


Fichier de notre choix:





1ère image: début de la formation du grid de communication.
2e image: les robots de prospection vont sous peu découvrir les gisements.



3e image: les robots forages sont sur les gisements et les robots de transport sont en action.