24-05-2021
Imane Jennane 310900
Daniel Elmaleh 311287

RAPPORT FINAL

- **ALGORITHME DE CONNEXION**

L'idée : Afin d'établir les connexions, nous avons créé un tableau de Robots pour chaque base. Si par le test de connexion nous obtenons qu'un robot est connecté à sa base, il alors ajouté au tableau de connexion.

Stratégie de parcours du graphe : Deep First Search (DFS) ou parcours en profondeur, algorithme récursif.

Initialement, tous les attributs booléens remote des Robots de Simulation sont mis à false au début de chaque mise à jour.
Ensuite on choisit de parcourir le graphe de connexion à partir de chaque robot et ainsi, créer un attribut de Robot : tableau de robots connectés au robot de départ.
Si en parcourant ce tableau, on a présence du Robot de Communication situé au centre de la Base du robot de départ, alors celui-ci (le robot de départ) est rajouté au tableau des robots connectés à la Base et son booléen remote est mis à true.
L'exploration s'arrête dès que tous les robots (ou sommets) de la simulation soient tous visités.

- **ALGORITHMES COMMUNICATION - PROSPECTION – FORAGE – TRANSPORT**

Algorithme de communication :

Nous avons trouvé meilleur de définir des robots de Communication en **mode Remote uniquement**, en définissant 8 directions de lancement selon le nombre total de robots de Communication et à une distance $n\ x\ rayon\ de\ communication$, n étant le *(nbRobotsCom-1) modulo 8 (le 1er robot de communication reste au centre de la base (cf. Rendu2)).

Algorithme de prospection :

Lorsqu'un robot prospecteur est en mode **Remote**, il avance en direction de son but initial (déterminé par sa base) mais avec un nouveau but à chaque fois, situé à une distance DeltaD par rapport au but précédent, afin de vérifier à chaque mise à jour de la simulation si sa position coïncide avec un gisement.

Lorsqu'il devient **Autonomous** avant que son booléen atteint soit mis à true, il continue tout simplement vers son but initial déterminé par sa base. Si au contraire, son booléen atteint est déjà mis à true, on lui attribue de nouvelles directions pour son avancement.

Algorithme de Forage :

En mode **Remote**, chaque fois que la simulation indique qu'il existe au moins un gisement dont la quantité de ressources est plus grande que DeltaR, alors un Robot Forage libre est envoyé vers le point le plus proche du gisement de sa position actuelle.
En mode **Autonomous**, il continue vers son but initial déterminé par la base avant son lancement et s'immobilise.

Algorithme de Transport :

Quand il est en **Remote** et dès que la simulation indique qu'un Robot Forage ne s'est pas encore débarrassé des ressources puisées dans un gisement, un Robot de Transport est envoyé vers la position du forage, récupère les ressources et ensuite s'oriente vers la base pour les lui donner.

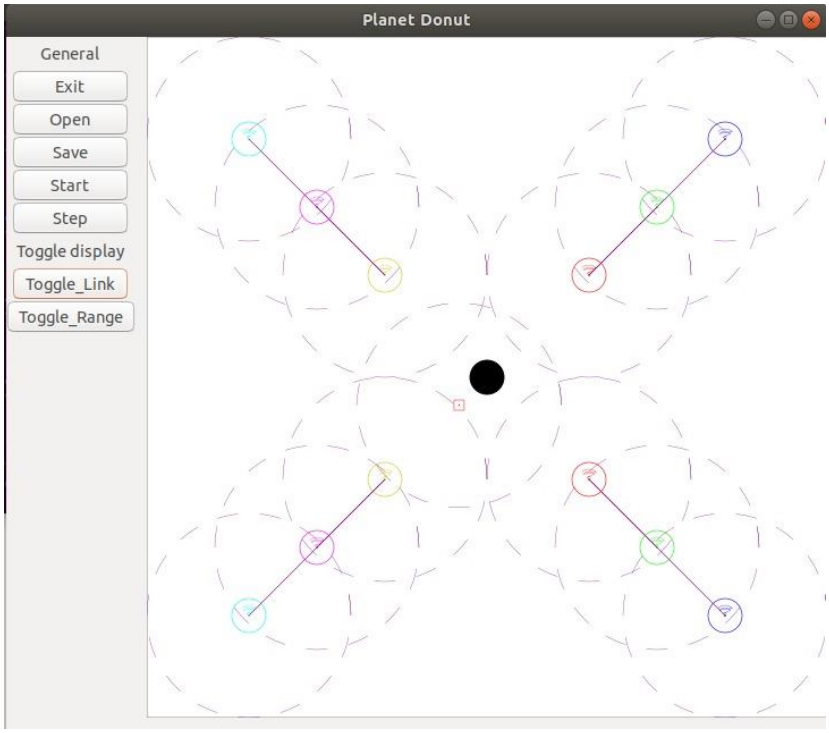
En **Autonomous**, si son booléen atteint est true alors il revient vers la position de sa base. Si atteint est mis à false alors il continue son chemin vers le but déterminé.

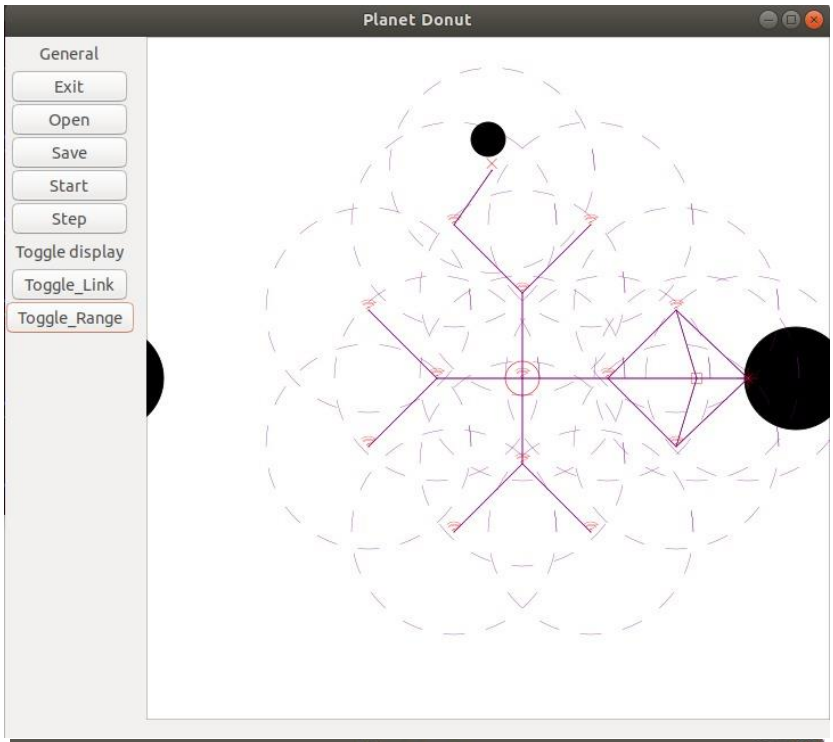
- **ROBUSTESSE DE L'APPROCHE PRÉSENTÉE**

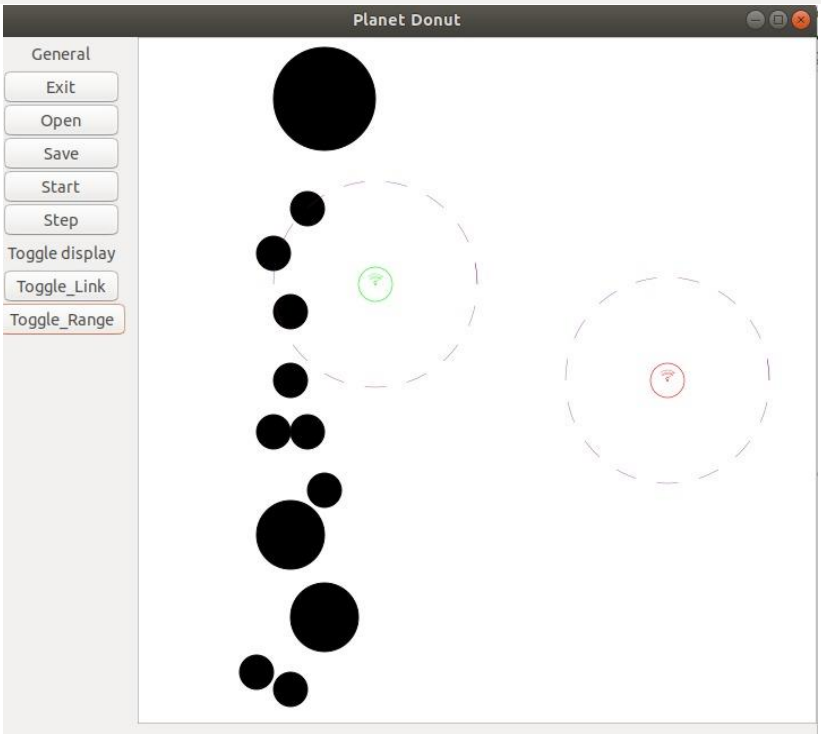
Ce qui permet à notre fonctionnement d'être plutôt optimal est que chaque robot suit une direction plutôt différente d'un autre, tout en essayant d'être à une distance qui lui permet d'être connecté avec sa base. Aussi il y a la stratégie du robot prospecteur, qui lui donne à chaque mise à jour comme nouveau but, la **direction** de son but initial (à la création) mais avec un pas de DeltaD, ainsi il pourra à chaque pas de la simulation renvoyer ses coordonnées à sa base afin qu'elle mette son found à true s'il est à proximité d'un gisement puis y envoyer un robot forage et un robot de transport.

- **ORGANISATION DU TRAVAIL**

Nous avons choisi pour le Rendu 3 de créer des méthodes et des fonctions dans un module à part afin de coder globalement toutes les idées que nous pensions être adaptées à la donnée du projet et pour ne pas apporter des changements inutiles dans les modules qui ont été mis au point précédemment, jusqu'à être sûrs de l'efficience de notre stratégie, chose qui nous a permis de choisir d'une manière plus optimale l'emplacement de chaque action.
Concernant la dispersion des taches, nous avons préféré de presque tout faire à 2.







Le bug le plus fréquent : Présence de segmentation fault : Résolu par l'utilisation des pointeurs.

Auto-évaluation : Nous avons essayé au maximum de suivre progressivement les étapes proposées par la donnée, ce qui nous a aidé à optimiser du temps, et nous nous sommes référés au cours pour bien comprendre comment aborder le projet (comme l'algorithme de connexion présenté en semaine 4 et les séries POP).
Sans oublier l'outil incontournable à ce projet et qui a résolu plusieurs de nos problèmes : **Discourse**.
Nous avons par contre été légèrement contraints au niveau du temps et donc pas eu l'occasion de mettre en place toutes les idées qui paraissaient très exploitables mais longues par rapport au temps dédié au projet.