Rapport 3  Nicolas PFITZER, Diane WIN

1. **Algorithmes :**

**Connexion :**

L’algorithme permettant d’établir les connexions fonctionne de manière récursive. En commençant avec le robot de communication, il permet de parcourir les listes d’adjacence des robots connectés à la base en changeant leurs booléens *visited* à true. Ceci permet d’établir une liste de tous les robots connectés à la base (y compris éventuellement les robots d’une autre base). Cette liste est ensuite parcourue afin de déterminer lesquels de ces robots appartiennent à la base qui fait l’appel de la fonction. On se retrouve donc finalement avec deux listes : *AllRoboConnex* (tous les robots connectés à la base) et *baseRoboConnex (seulements les robots de la base en question).*

**Prospection :**

Les robots de prospection sont uniquement créés au début de la simulation (si la base a assez de ressources). Nous en créons 10 car nous estimons que cela est suffisant : ils sont réutilisés jusqu’à la fin de la simulation grâce à des maintenances fréquentes.

Chaque fois qu’un robot de prospection atteint son but, celui-ci est redéfini. Il se dirige vers sa base s’il est en *autonomous* et qu’il a atteint son but (peu importe qu’il ait trouvé un gisement ou non), ou s’il lui reste peu de distance à parcourir (nous estimons que lorsqu’il ne peut parcourir que 2 x x *dim\_max*, ou moins, il doit revenir à sa base pour faire une maintenance, car c’est la plus longue distance parcourable dans l’espace avec le rebouclement).

S’il est en *remote*, son but est redéfini selon un simple algorithme qui utilise des simples calculs de trigonométrie afin de changer son orientation.

Son booléen found passe à true lorsqu’il trouve un gisement, uniquement si celui-ci n’a pas déjà été trouvé par un autre robot de prospection (pour cela nous initialisons un booléen *used* dans chaque gisement, qui passe à true lorsqu’un robot de prospection le trouve).

Nous estimons que la maintenance des robots de prospections est utile, dans la mesure où le coût de celle-ci est toujours plus basse que si on décidait de créer un nouveau robot.

**Forage :**

Les robots de forage sont créés lorsqu’un robot de prospection a trouvé un gisement (son booléen *found* est true) et est en *remote* (et si la base a assez de ressources pour créer un robot de forage et un robot de transport).

Dès sa création, il se dirige vers le gisement pour lequel il a été créé et s’arrête juste à l’intérieur de celui-ci (à 1 unité du bord) et reste ici pour toujours. À son arrivée, un de ses attributs, un pointeur de gisement *gis*, s’initialise pour pointer sur ce gisement.

Dans la mesure où il ne bouge plus dès que son but est atteint et qu’il a un seul et unique but lors de toute sa vie, on n’effectue jamais de maintenance dessus.

**Transport :**

Les robots de transport sont créés en même temps que les robots de forage. Nous avons décidé de limiter le nombre de robots créés à 4, car ils sont très coûteux et peuvent être réutilisables dans la mesure où la base effectue une maintenance sur eux.

Après être arrivé à son premier but (le même que le robot de forage créé en même temps que lui), et qu’un forage a été effectué, le robot de transport revient vers sa base pour ramener *deltaR* ressources. À ce moment-là, son but est redéfini par sa base : un algorithme parcourt les robots de forage (en *remote*), regarde si leur *gis* pointe sur un gisement, et si ce gisement est vide ou non. Ensuite il compare tous les gisements vers lesquels il pourrait aller et se dirige vers celui dont le robot de forage est le plus proche (son but est donc la position du robot de forage qui fore ce gisement).

À chaque fois qu’un robot de transport revient à sa base, une maintenance est effectuée. Les robots de transport sont les plus chers (leur coût est de 100 ressources), donc effectuer des maintenances au lieu de les remplacer est plus efficace. Aussi, ils reviennent forcément à leur base régulièrement pour acheminer les ressources, donc il est pratique de faire une maintenance à ce moment-là.

**Communication :**

Les robots de communication sont uniquement créés au début de la simulation, dans le but de créer une grille sur l’intégralité de l’espace pour que tous les robots soient en *remote* et que nos stratégies soient plus efficaces. Pour créer une grille effective, nous utilisons 49 robots.

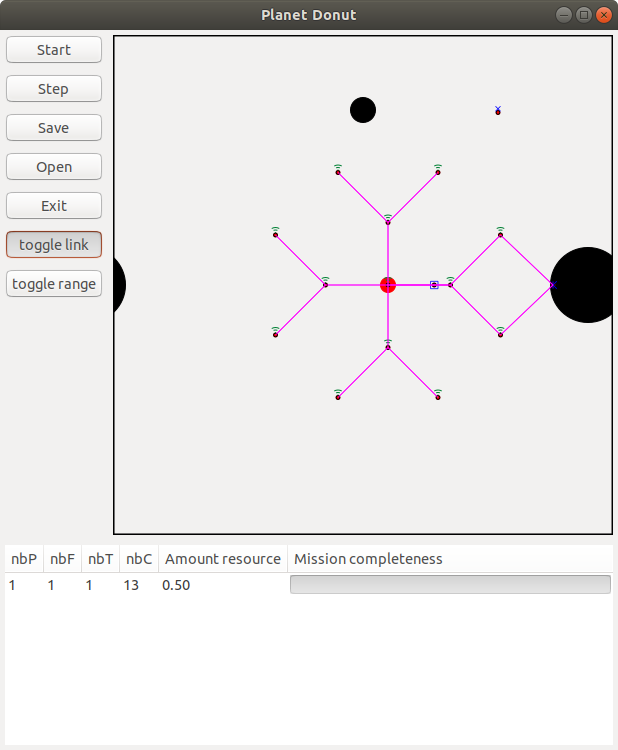
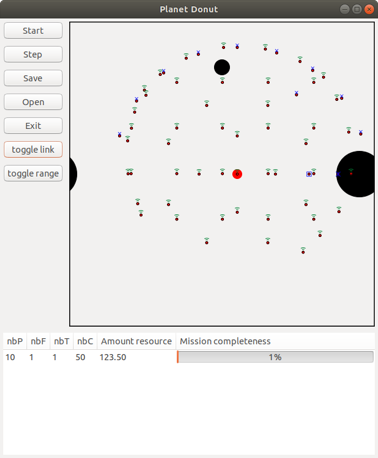
Leur but est uniquement de créer une grille de communication. Leur but est ainsi défini dès le départ et ne change pas en cours de route.

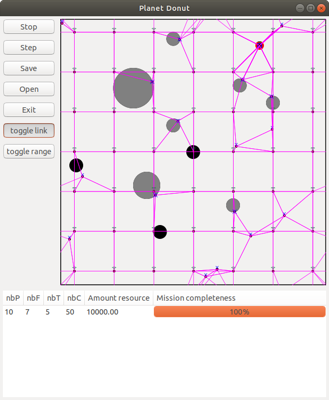
Ces robots n’ont qu’un seul but au cours de la simulation et ne bougent plus dès qu’ils l’ont atteint, donc aucune maintenance n’est effectuée sur eux.

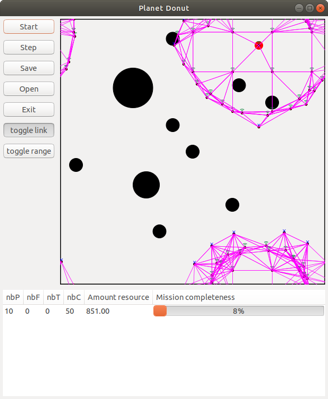
Notre approche est robuste dans la mesure où nous gardons les dépenses en ressources de la base assez faible : peu de robots de transports sont créés et il n’y a jamais 2 robots de forages sur un même gisement, et nous préférons effectuer des maintenances à recréer des robots, lorsque possible.

De plus notre algorithme de recherche de gisement, via les robots de prospections, est efficace car ceux-ci ne parcourent pas de très grandes distances (donc ils vérifient plus régulièrement s’ils ont trouvé un gisement) tout en changeant de direction à chaque fois que leur but est atteint, ce qui permet d’aller partout sur l’espace de la planète.

1. Captures d’écran



Une image contenant texte, intérieur, capture d’écran

Description générée automatiquement

1. **Méthodologie**

Afin de bien répartir le travail, nous avons commencé par diviser se dernier rendu en deux parties : L’aspect graphique et la stratégie/le comportement des robots.

**Taches de Diane Win***(modules : base/robot)* **:**Je me suis occupée du code en lien avec le comportement des robots. Le défi majeur était de trouver une stratégie robuste qui permettrait d’obtenir le succès des bases dès que les ressources disponibles dans les gisements étaient suffisantes tout en respectant l’encapsulation et l’accès à l’information des différentes entités.

**Taches de Nicolas Pfitzer***(modules : gui/graphic)* **:**

Je me suis tout d’abord chargé de la partie graphique de ce dernier rendu, il fallait être capable d’assimiler une grande quantité de nouveaux concepts liés à GTKM et de pouvoir les appliquer dans le contexte de ce projet. Comme cette étape était relativement courte, je me suis ensuite occupé de l’écriture de fichiers.

Une fois nos taches respectives terminées, nous avons ensuite travaillé ensemble afin de combiner, adapter et surtout débugger notre code Ceci a fini par être la partie la plus longue et intensive de ce projet.

Il est difficile de définir qui était en charge des autres modules, car nous avons tous les deux travaillés dessus mais a des instants différents. De manière générale, nous avons suivis les directions des données de chaque rendu pour l’ordre de conception des modules.

Le bug le plus fréquent était *le segmentation fault* dû à l’utilisation de pointeurs. Les pointeurs nécessitent d’être très méthodique lors de l’écriture du code et la moindre erreur ne pardonne pas. Ceux-ci ont causé en particulier causés beaucoup de problèmes lorsque nous avons commencé à utiliser des constructeurs de copie, car la copie de pointeurs nécessite une attention supplémentaire.

1. **Conclusion**

Ce projet, bien que très long, nous a beaucoup appris sur la conception d’un programme et le travail de groupe. La dynamique de groupe était très bonne et la charge de travail était bien répartie. Ne connaissant cependant pas les outils/programmes permettant le partage de code dans un groupe, nous réalisons maintenant que notre travail aurait cependant pu être beaucoup plus efficace si nous avions pris le temps de faire des recherches sur un meilleur moyen de transférer le code entre binômes.