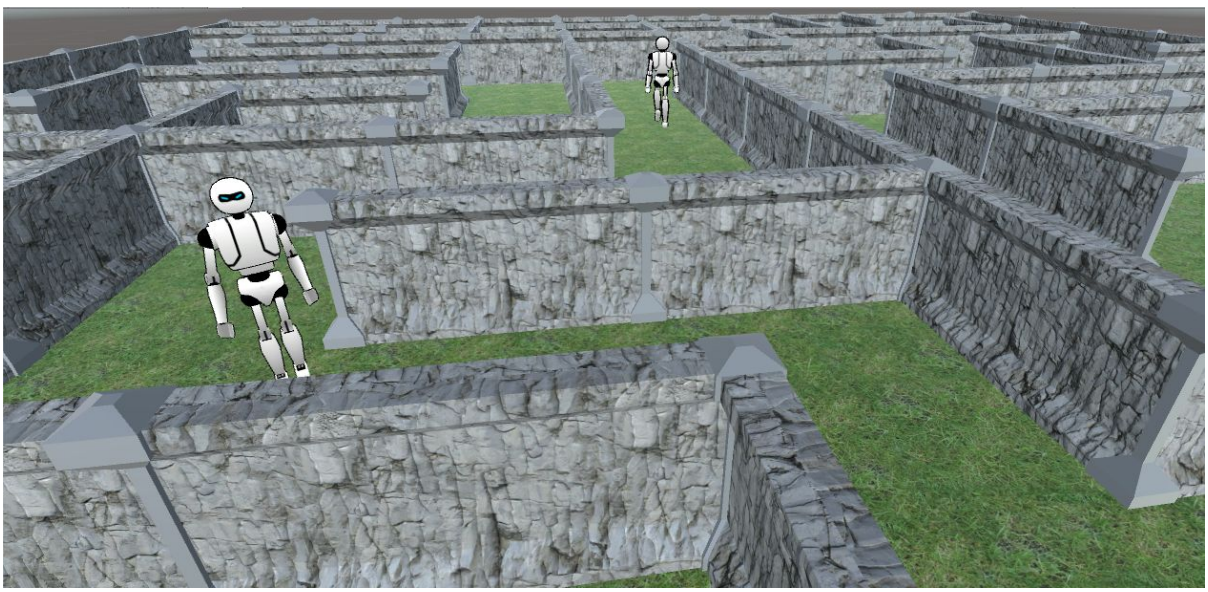
Systèmes multi-agents

Rapport d'activité

Encadrant :
Pierre Alexandre Favier

Élèves :
Martin KHANNOUZ
Damien SUREAU



Présentation du projet

Ce projet s'inscrit dans le cours de Systèmes Multi-Agents dispensé à l'ENSC dans le cadre de la formation en robotique commune à cette école et à l'ENSEIRB-MATMECA. Les systèmes multi-agents constituent une approche de la résolution de divers problèmes par la simulation des interactions entre des "agents", entités possédant un comportement simple, dans un environnement adéquat. Cette méthode permet de répondre à des problématiques pour lesquelles il n'existe pas de solution formelle, ou bien si celle-ci est trop complexe. Bien que les résultats obtenus soient fiables pour la plupart, ils sont sensibles aux différents paramètres de la simulation et ne constituent jamais qu'une approximation.
Le but de ce projet est alors d'appliquer cette méthode à un cas particulier, puis d'en tirer les conclusions sur l'efficacité de cette approche les ressources qu'elle demande. Dans le cas que nous traiterons ici, il s'agira de comparer deux méthodes de découverte d'un labyrinthe par une multitudes d'acteurs qui peuvent communiquer entre eux. Le premier cas d'analyse demande aux agents d'explorer la carte en retournant régulièrement à une base pour stocker leur découvertes, quand le second cas permet au agents de communiquer entre eux lorsqu'ils se rencontrent. Le but de la simulation est de déterminer l'efficacité de chaque cas en temps d'exploration et de pouvoir comparer les résultats obtenus selon la taille de la carte à découvrir et la méthode employée.

Axes de recherche

Cas 1 : Retour à la base

Ce premier cas oriente la simulation vers une gestion centralisée de la connaissance du terrain à découvrir. Si chaque robot connaît effectivement les cases qu'il a parcouru, il devra partager cette connaissance en revenant régulièrement à la base. Cette base contient un objet appelé Nexus dans la simulation, et qui stocke la connaissance de la carte que lui rapportent les robots. En échangeant avec ce Nexus, chaque robot synchronise ses connaissances avec celles du Nexus, ce qui permet lui permet de faire évoluer la connaissance collective du terrain et de repartir éventuellement avec plus de connaissances qu'il n'en avait avant.
Dans ce cas, les robots ne peuvent pas communiquer directement entre eux et sont donc tenus de revenir à la base pour augmenter la connaissance gloable.
Cette approche permet de centraliser les connaissances est d'assurer que chaque robot bénéficie régulièrement de la connaissance commune, mais en obligeant des déplacements supplémentaires pour revenir régulièrement à la base.

Cas 2 : Base mobile

Ce second cas offre une approche décentralisée de l'exploration où les robots vont vaquer dans l'espace en essayant de tout découvrir. Occasionnellement, il va croiser l'un de ses

semblable et les deux compères vont pouvoir s'échanger leur connaissances respectives de la carte.
Ce cas se distingue du premier dans le sens où il n'existe pas d'entité de référence permettant de centraliser toute la connaissance du terrain. Il y a cependant deux types de comportement pour les robots, qui peuvent explorer ou bien essayer de faire le lien entre les explora teur en se dirigeant vers eux. Cette approche permet d'obtenir une découverte assez efficace puisque tous les éléments sont en mouvement et contribuent à la découverte du terrain. En contrepartie, les robots se déplaçant d'explorateurs en explorateurs sont moins efficaces dans leur tâche de découverte du terrain puisqu'ils ne passent que par des cases qui ont déjà été parcourues.
Afin d'éviter que tous les robots ne se rassemblent au même endroit, ils s'échangent, quand ils le peuvent, les cases qu'ils ciblent. Si deux robots se retrouvent à cibler la même case, le plus éloigné des deux changera de cible.

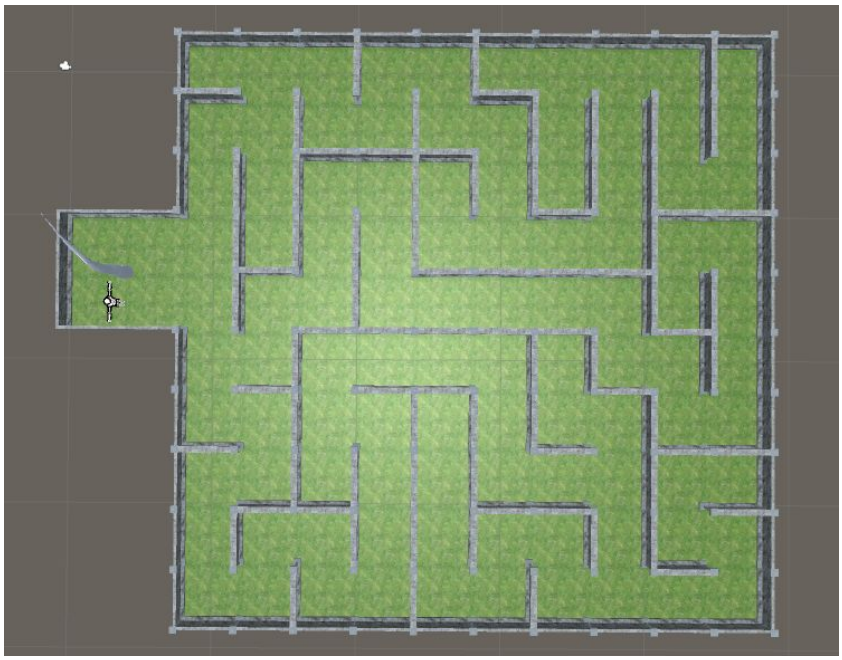
Condition d'arrêt

La simulation se termine lorsque tous les robots ont découvert la totalité de la carte. Dans le premier cas, un robot ayant terminé sa recherche retourne à la base puis s'y arrête. Cela permet de faire bénéficier les autres de cette connaissance, et le temps de la simulation restant correspond au temps qu'il faudra aux autres robots pour revenir à la base. Dans le second cas, les robots qui ont terminé leur recherche s'arrêtent à leur position courante et attendent que les autres aient terminé. Cette démarche permet encore de communiquer aux autres robots qui passent à proximité.

Implémentation

Génération de la carte

Les différentes cartes sur lesquelles nous travaillons sont des labyrinthes générés en amont des tests et modifiés pour correspondre aux cas d'études. Ces labyrinthes sont carrés et ne contiennent pas de cycles, de telle manière qu'un robot est obligé de revenir sur ses pas pour visiter des cases qu'il ne connaît pas encore. Cette contrainte est imposée par la méthode de génération que nous avons décidé d'utiliser, et peut augmenter le globalement le temps de convergence de notre simulation. Cependant, comme cette contrainte est présente dans les deux cas que nous testons, cela devrait assez peu impacter nos résultats relatifs.



*maze_1 vu de dessus - taille : 41*41*

Navigation et algorithmes de déplacement

Dans le but de simplifier le calcul des chemins au sein du labyrinthe, nous avons choisi d'utiliser un objet de type "navMeshAgent" attaché à chaque robot devant se déplacer sur le terrain. Cet objet permet de calculer automatiquement le chemin entre deux points sur la carte : la position courante du robot et sa destination. Notons que le fait qu'il n'existe pas de boucle dans les labyrinthes que nous utilisons permet de réduire le nombre de chemins possibles entre deux points à un, ce qui peut améliorer les performances de la simulation étant donné que cette méthode est appelée régulièrement.
En ce qui concerne le choix des destinations, chaque robot va utiliser ses connaissances pour déterminer la position qui est le plus proche de lui et qui lui est inconnues. Il est à noter que ces distances sont calculées en ligne droite et peuvent donc amener le robot à parcourir un grand trajet entre deux positions successives. Si cela ne favorise pas le parcours d'une branche du labyrinthe en peu de temps, cela allonge le parcours de chaque robot entre deux retours à la base et donc potentiellement le nombre de cases trouvées.

Découverte de la carte

La carte est découpée en une grille de même taille que le labyrinthe, pour obtenir des cases de taille 1*1. Cette grille est calculée avant le lancement de la simulation par un script attaché à la carte, il en ressort un masque contenant les positions des obstacles sur la carte et auxquels les robots peuvent accéder. Chaque robot possède une matrice de même taille que le masque, et peut la remplir à partir des éléments du masque au fur et à mesure qu'il parcourt la carte. En pratique, les robots ont une "vision" représentant un carré autour de leur position et ont donc accès à plusieurs cases à chaque frame.

Cas 1

Dans le premier cas, chaque robot parti en exploration se choisit un nombre fixe de destinations (fixé à 5 lors des tests) puis retourne à la base pour y synchroniser ses connaissances. Cela entraîne globalement que la majorité des connaissances de chaque robot est constituée de cases qu'il a lui-même découvertes, mais se sert de la connaissance globale qu'il a acquise auprès du Nexus pour éviter les cases déjà répertoriées.
Cette méthode permet de faire croître la connaissance collective assez rapidement au fur et à mesure que les robot reviennent à la base.

Cas 2

Dans ce second cas, les robots initialisent un tableau de la taille de la carte en spécifiant chaque case comme étant inconnue. Ensuite, son objectif sera de découvrir la case inconnue la plus proche de lui. Pour s'y rendre il utilisera le « NavMeshAgent » comme décrit plus haut.
Lorsqu'il rencontre un autre robot suffisamment proche, ils se partagent mutuellement les cases qu'ils ont découvertes et que l'autre n'a pas encore rencontrées.
La carte est déclarée complète lorsque qu'aucune case inconnue ne se trouve à côté d'une case sur laquelle on peut marcher. Cela se traduit par le fait que l'on a visité toutes les cases accessible depuis la position initiale.
Après plusieurs tests sur de grande cartes moins rempli qu'un labyrinthe, nous nous sommes rendu compte que cette seconde méthode pouvait prendre beaucoup de temps. En revanche, nous avons remarqué que la connaissance collective croissait lorsque que les robots se croisait régulièrement.
À défaut d'avoir imaginé un algorithme complexe pour les faire se croiser plus souvent, nous avons choisi d'introduire un seconde type de robot dont la seule raison de vivre était de circuler entre tous les autres robots. Cela permet de mieux ventiler les connaissance de la carte en essayant d'atteindre les robots les plus éloignés.

Communication

La communication entre plusieurs entités, à lieu par l'appel des méthodes auxquelles chaque entité peut accéder au sein du script de l'entité duale dans la communication. Ces méthodes effectuent une comparaison des deux cartes et affectent les cases différentes.

Résultats

Critères d'évaluation

Notre critère d'évaluation principal et le temps de la simulation sur une carte fixée avec un même nombre de robots. Ainsi, nos tests ont été effectués avec 3 cartes de taille 41*41 et 3 cartes de taille 33*33. Le nombre de robots est fixé à quatre dans les deux cas.

Comparaison des deux méthodes

Après des simulations sur les différentes cartes et dans les conditions citées ci-dessus, nous obtenons les résultats suivants :

Maze	**cas 1**	**cas 2**
maze_1	260	195
maze_2	347	202
maze_3	303	187
maze_4	201	133
maze_5	180	106
maze_6	174	136

Chaque valeur représente le temps en secondes mis pour chaque simulation. On peut remarquer globalement que le cas 2 est plus performant que le cas 1. Cela est sûrement dû à la différence de distance à parcourir par chaque robot. Il est également possible de remarquer que les labyrinthes 4 à 6 sont plus rapides que les premiers, ce qui est logique étant donné qu'ils correspondent aux cartes de tailles plus petites.