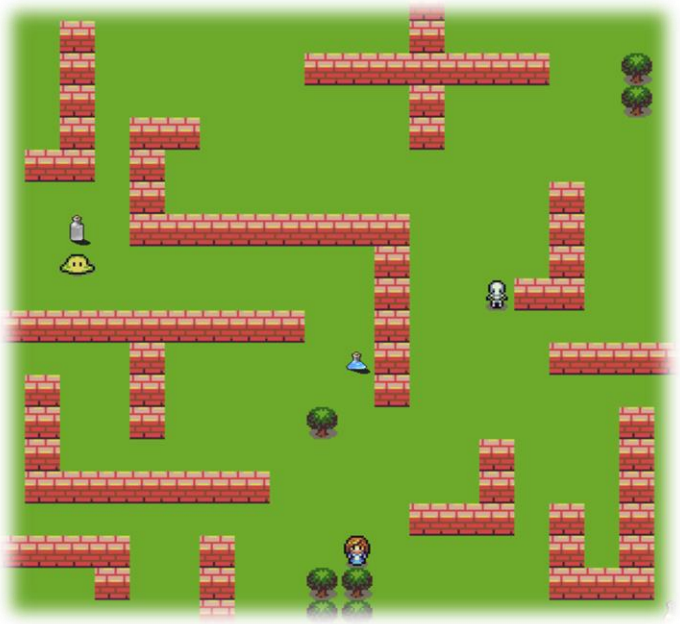

Mini-projet (2019): Cooperative Path-Finding

LY Kimmeng / JIN Yang-kang

Nicolas Maudet
3I025

Le projet consiste à appliquer différentes notions vues en cours telles que les heuristiques comme la distance de Manhattan, ou bien appliquer l'algorithme A* sur différentes unités (Ici des personnages/joueurs) pour leur permettre de récupérer des objets (fioles) à partir de différentes stratégies imposées.

Par la suite, on va vous présenter les trois stratégies implémentées en commençant par la Stratégie Opportuniste, puis la Stratégie coopérative de base et enfin finir par la Stratégie coopérative avancée.

Nous avons, dans un premier temps, implémenté l'algorithme A* de base avec un seul joueur sur la carte pour pouvoir le réutiliser par la suite dans les différentes stratégies à implémenter.

L'initialisation est la même pour chaque stratégie. Elle diffère selon les cartes chargées. C'est-à-dire que le nombre de joueur dépend du nombre de joueur sur la carte etc…

Nous avons 3 listes d'initialisation qui vont nous servir de base :

initStates : contient la localisation initiale de tous les joueurs présent sur la carte

goalStates : contient la position tous les objets ramassables

wallStates : contient la position de tous les obstacles

I. Stratégie Opportuniste de portionnement de chemin :

Le but de cette stratégie est lorsqu'il y a un conflit (croisement avec un autre joueur), on recalcule alors une partie du chemin via A*, afin de contourner cet obstacle « dynamique ».

Pour implémenter cette stratégie, on a calculé les chemins des différents joueurs avec A*. Les objets à ramasser, sont attribués aléatoirement pour chaque unité.

Avant de déplacer chaque joueur, on récupère leur position suivante puis on vérifie s'il n'y a pas un autre joueur à cette position.

S'il y a un autre joueur à cette position, on ajoute cette dernière position à la liste des obstacles (wallStates) puis on recalcule A* à partir de la position courante (mais dès lors que le joueur « obstacle » n'est plus à cette position, on la retire du wallStates), l'objet à ramasser reste inchangé mais la liste des obstacles est mise à jour en prenant compte de l'obstacle « dynamique » (Joueur). Une fois l'objet ramassé le joueur ne bouge plus, il reste sur sa position finale. Une fois que tous les objets sont ramassés, on affiche le score de chaque joueur.

II. Stratégie Coopérative de base

La stratégie Coopérative de base consiste à identifier les chemins des différents joueurs ne partageant aucune case. Il s'agira alors de choisir les joueurs ayant des chemins différents pour éviter une collision. Si deux joueurs partagent au moins une même position dans leur chemin, ils ne doivent pas se déplacer en parallèle, l'un d'eux devra récupérer son objet avant que l'autre puisse se déplacer à son tour.

Comment avons-nous procédé ?

On a d'abord calculé les chemins des différents joueurs en leur adressant un objet à ramasser aléatoirement.

Maintenant que l'on a nos chemins (Path), on a créé une liste (non_croisement) dans lequel on a créé x liste, x étant le nombre de joueur (nbPlayers).

On compare alors la première liste avec toutes les autres listes, on fait la même procédure avec la deuxième liste jusqu'à la dernière liste etc…

Si on a des joueurs ayant des chemins différents on ajoute dans leur liste respective le joueur avec lequel il pourra partir sans se croiser.

Ensuite on trie la liste non croisement selon la longueur (len) de ses éléments, du plus petit élément (à l'index 0) au plus grand élément.

On va itérer sur le premier élément de la liste non_croisement qui contient les joueurs qui peuvent se déplacer sans se croiser, on va déplacer ces joueurs séquentiellement jusqu'à ce qu'il atteigne leur but.

On supprime alors dans les autres listes de non_croisement, les joueurs qui se sont déplacés pour qu'il ne puisse plus se déplacer à nouveau.

On déplacera les joueurs restants par la suite en suivant la même procédure.

III. Stratégie Coopérative Avancée

Pour cette stratégie finale, on a pris comme base la stratégie précédente. Nous avons modifié le A* en ajoutant une 3 -ème variable t qui est le temps en plus de x,y qui correspondent à la position. Ainsi on peut connaître d'avance à quel itération le joueur sera dans une position donnée. Donc même si deux joueurs ont des positions communes sur leur chemin respectif, si le moment t des deux joueurs sur cette position x,y est différent, les deux joueurs pourront

se déplacer parallèlement. Sinon s'ils ont une même position aux même moments t, ils attendent et se déplaceront chacun leur tour.

IV. Comparaison des stratégies

En temps d'exécution :

Nous avons mesuré le temps d'exécution de chacune des stratégies sur une même carte et nous avons pu constater que la stratégie ''path slicing'' met le plus de temps à s'exécuter (5.04 secondes), puis vient la stratégie de coopérative de base (5.03 secondes) et donc la stratégie qui met le moins de temps à s'exécuter est la coopérative avancée (4.84 secondes).

Au nombre d'itérations :

Si on compare en nombre d'itération la stratégie coopérative avancée reste la mieux placée (35 d'itérations en moyenne) alors que les autres stratégies ont besoin de 55 d'itérations en moyenne pour ramasser tous les objets.

Conclusion :

Les résultats obtenus sont ceux qu'on attendait. La stratégie coopérative avancée a été optimisée à partir des deux autres, elle est en effet la stratégie la plus rapide.

A travers ce projet nous avons pu appliquer certaines notions du cours et découvrir comment fonctionne les agents dans les différentes stratégies implémentées.