Groupe 2E : Département Informatique
RAVIART Alexandre, DROUCHEAU Quentin, SEVELLEC Maxime

Rapport SAE SPLAT IUT'O

Fonctions implémentées et stratégie de l'IA

Stratégie de l'IA :

L'IA s'articule autour de 7 états flexibles selon les données du jeu (données des joueurs, données de la partie, données du plateau). Ceux-ci s'activent sous certaines conditions pour optimiser au mieux les décisions parmi les tours que l'on dispose.

L'état « start » : Pour initialiser son territoire de peinture, le joueur va au départ chercher la meilleure position et le meilleur angle de tir pour répandre le plus de peinture sur une distance.

L'état « stack » : Lorsque la réserve du joueur est inférieur à 0, l'état est à la valeur « stack » qui répond au besoin de recharger sa réserve, le joueur va alors se déplacer vers deux cases de la même couleur que lui et à proximité, puis alterner ses déplacements entre ces deux positions afin de gagner une réserve de façon passive.

L'état « objet » : À condition que le coût de l'itinéraire vers l'objet soit inférieur à celui de la réserve, l'objectif de cet état est de mener une offensive grâce à la puissance de la bombe et du pistolet principalement (un système de priorité entre objets existe).

L'état « attaque » : À condition que la réserve du joueur soit supérieure à 20, que le coût pour aller à la case ennemie est inférieur à celui de la réserve et qu'il ne possède pas d'objets, on a alors la réserve nécessaire pour mener une offensive, le joueur va chercher les cases de couleurs adverses le plus proche pour les peindre.

L'état de « bidon » : C'est en cas d'urgence, si le joueur ne possède plus de case de sa couleur et que sa réserve est négative, le moyen de monter sa réserve est le bidon qu'il cherchera jusqu'à l'obtention.

Respectivement l'état « pistolet » et « bombe ». Tous deux vont pousser le joueur à une position où ils trouveront la meilleure configuration de tir : une rangée dont la distance de tir recouvre un maximum de mur pour le pistolet et un rayon de 1 case dans au minimum 3 directions pour la bombe. Ainsi on tire profit au maximum des objets.

L'IA a donc une stratégie d'expansion par pression offensive lors de l'utilisation efficace d'objets, et en cas de mauvaise posture elle adopte une stratégie de « farming ». Ainsi de suite.

Fonctions implémentées et complexité :

Concernant les principaux algorithmes qu'on peut retrouver dans cette IA :

On peut trouver à la base de beaucoup de stratégies la fonction qui fabrique la calque, elle retourne un calque du plateau indiquant la distance de chaque case par rapport à la position d'un élément. Il s'agit

d'une boucle imbriquée avec un appel de fonctions de directions possibles au sein du 2ème for, qui contient elle-même une boucle. On a donc O(N)*O(N)*O(N) (le reste est négligeable) = $O(N)^3$ soit un grand coût.

Celle qui détermine la direction que le joueur doit prendre en fonction de 2 positions : la fonction direction_chemin(plateau_jeux, pos1, pos2) qui retourne la direction à prendre afin d'aller a la deuxième. Elle contient une boucle while et un for, mais surtout, un appel a calque_plateau (qui peut être évité en passant direct en argument celui-ci depuis l'environnement de la fonction mon_IA(), il s'agit donc d'une complexité O(N)* O(N)* O(N)* O(N) = $O(N)^4$ qui pourrait être de complexité $O(N)^2$.
De l'autre côté, grâce à une structure de données par dictionnaire : lorsqu'il s'agit de récupérer une donnée depuis le plateau ou de vérifier si telle donnée est contenu dans la clé du plateau ou du calque, la complexité est de O(1).

Difficultés

La principale difficulté à été la répartition des tâches lors du développement de l'IA. Au départ nous avions comme projet que chaque membre du groupe réalise sa propre IA et qu'ainsi on puisse les comparer en les affrontant. Mais le développement fut plus long et surtout il était plus rentable de mettre en commun notre travail au milieu de la phase de développement (fonctions annexes et autres structures) pour avoir le temps d'avancer avec un point de vue commun. Nous avons finalement fusionné notre travail avec le membre avec l'IA la plus développée pour, à la fin, avoir l'IA que nous souhaitons.
Enfin, pour ce que nous aurions pu faire ou chercher à faire, c'est la mise en place d'une ia à apprentissage automatique ou nous aurions entraîner un modèle des milliers de fois sur des milliers de mouvements avec des bibliothèques python tel que Tensor Flow ou Scikit-learn et ainsi prendre les meilleurs décisions possible.