Diamant IA

Rapport:

La première piste que nous décidons d'explorer est la suivante:
Faire une IA à qui on fournit toutes les informations possibles sur la partie en cours, et qui à partir de ces informations va calculer trois indices différents:

- La dangerosité, qui selon le nombre de pièges et le nombre de cartes retournées, calcule le pourcentage de chances que la prochaine carte retournée soit un piège qui provoque la fin de la manche.
- La rentabilité à rester, qui selon le nombre de joueurs encore en exploration, et le nombre de diamants déjà retournés, calcule le nombre moyen de rubis qu'elle aurait à gagner en continuant l'exploration à la manche suivante.
- La rentabilité à rentrer, qui selon le nombre de joueurs encore en exploration, les diamants n'ayant pas été récupérés, les reliques retournées et le nombre de rubis amassés au cours de la manche calcule le nombre de rubis qu'elle aurait à gagner et à mettre en sécurité dans le coffre en rentrant.

A partir d'une moyenne de ces indices, elle va faire un pourcentage de chances entre continuer ou arrêter l'exploration et prendre sa décision avec ce pourcentage.

Notre but est ensuite d'effectuer de nombreux tests afin de déterminer quel pourcentage est le plus efficace pour notre IA.

Informations à fournir à notre IA:

- Le nombre de cartes retournées au cours de la manche actuelle
- A quelle manche en somme nous
- Le nombre de joueurs
- Un calcul du nombre de rubis de chaque joueur au fur et à mesure de la partie
- La liste des cartes ayant été retournées au cours de la manche
- La liste complète de toutes les cartes figurant dans la pioche

Après de nombreux tests, nous nous sommes rendu compte que l'indice "rentabilité à rester" n'était pas pertinent et que les IA qui ne le prenaient pas en compte avaient de meilleures performances que les autres. Nous avons donc totalement retiré de notre code cet indice.

Une fois les indices créés, nous avons dû estimer leur importance dans les choix de notre IA. C'est-à-dire que nous avons utilisé des coefficients pour faire une moyenne avec les indices (Danger et rentabilité à rentrer). Nous avons donc fait s'affronter environ 15000 d'affilée 4 IA en modifiant petit à petit les coefficients pour savoir laquelle est plus performante, puis on conservait les paramètres de l'IA la plus performante.

Nous avons aussi modifié le nombre par lequel on divisait la moyenne et nous sommes rendu compte qu'en le diminuant légèrement, l'IA performait bien mieux.

Nous avons ensuite décidé de tester notre IA envers des IA d'autre groupe et on remarque très vite que notre IA à le bon niveau requis.

Readme Diamant

Côme Regnier Cyril Tilan

Le but de la SAE était de programmer le jeu de société Diamant. Tout d'abord avec une version fonctionnant dans le terminal, puis dans une version graphique à l'aide du module Tkiteasy. Nous avons codé à l'aide de la fonction "code with me" de PyCharm qui permet de travailler à plusieurs en même temps sur la même page de code

Pour lancer notre jeu en version graphique, il suffit de lancer le programme , et la fenêtre tkiteasy s'affichera. Celle-ci est de base sur la résolution 900 par 700 pixels mais elle peut être adaptée à n'importe quelle taille. Il faut ensuite choisir le nombre de joueurs en cliquant sur le nombre désiré et la partie va pouvoir commencer. Les cartes sont affichées en haut et au milieu de l'écran, il y a les joueurs, ainsi que le nombre de rubis qu'ils ont dans leurs coffres.
Pour le choix des joueurs a chaque tour, le carré vert signifie que le joueur continue l'exploration et le rouge signifie qu'il souhaite rentrer au campement.

Pour lancer notre jeu sans la version graphique, le procédé est à peu de choses près le même. On double clique sur "diamant_terminal.py" et magique, le jeu se lance sur le terminal de commande. Au fur et à mesure de la partie, les coffres ainsi que les cartes retournées sont affichées dans le terminal et les différents joueurs doivent choisir s'ils continuent l'exploration ou non en tapant 1 ou 0 dans le terminal.

Voici la manière dont nous avons accompli notre travail:

La première étape a été de réfléchir à la structure du programme ensemble, et de déterminer la manière de stocker les différentes données ainsi que les fonctions que nous allions faire.
Nous avons ensuite écrit le programme principal appelant les fonctions, ainsi que les docstrings des fonctions elles-mêmes.
L'étape d'après était de procéder à l'écriture du programme à l'intérieur des différentes fonctions, avec bien sûr des tests à chaque fois.
En ajoutant quelques print des coffres et des cartes retournées, notre programme était désormais jouable en ligne de commande.
Il fallait désormais que nous nous attaquions à la partie graphique. Nous avons préparé nos fonctions d'affichage dans un autre fichier afin de ne pas être gêné puis nous les avons copiés une à une dans le programme écrit auparavant. Il ne restait plus qu'à faire quelques ajustements et notre programme était fin prêt.

Grâce à une bonne préparation en amont, nous n'avons pas rencontrés beaucoup de problèmes; mais certaines fonctions nous ont tout de même faites faire face à des difficultés:

- La fonction de retour des joueurs, car il y avait beaucoup de choses à gérer comme la récupération des reliques, la distribution des rubis… Nous avons passé beaucoup de temps sur cette fonction car elle ne distribuait pas correctement les rubis lorsque plusieurs joueurs rentraient en même temps. Le problème venait de notre appel de fonction, dans lequel on donnait un mauvais paramètre.

- La fonction qui demandait si les joueurs voulaient continuer à explorer ou s'ils voulaient rentrer. Il a fallu trouver un moyen de faire en sorte que la fonction ne redemande pas aux joueurs déjà rentrés le choix qu'ils souhaitaient faire. Ce problème a conduit à l'ajout de la liste " choix_definitif ".
- La fonction d'affichage des cartes retournées en version graphique. L'affichage des cartes en lui-même n'a pas été embêtant, mais ce qui nous a posé souci, c'est lorsqu'il fallait supprimer ces cartes parce qu'un nouveau tour commençait. La solution que nous avons trouvé est de placer chacune de nos cartes dans une liste, puis de supprimer toutes les cartes de cette liste dès que le tour se finissait.
- Même souci avec l'affichage des coffres des joueurs.
- Le fait de centrer les textes quelque soit la résolution de l'écran a été fortement compliqué du fait qu'il faut diviser la largeur de la fenêtre par un chiffre assez précis.