Résumé

Équipe Tatou

30 mars 2021

Après de nombreuses nuits de dévelloppement acharné, l'équipe Tatou est fière de vous présenter le Tatou-robot ultra sophistiqué. Pour réaliser cette prouesse technologique, nous avons codé l'intégralité du code sous Éclipse, intégrant le plugin eGit qui nous a été d'une grande utilité pour le travail collaboratif.

La premiere étape fut l'ogranisation d'une réunion afin de mettre en relation ce que chacun avait saisi du sujet, de façon à ce que tout le groupe parte sur une même compréhension du sujet. De plus, nous avons commencé à réfléchir à la conception de notre projet et la réalisation de diagrammes UML. Suite à cela, nous avons implémenté les classes dans différents packages, en fonction du diagramme de classes que nous avions réalisé. Notre code est divisé en 5 packages. Le premier, equipements, c'est ici qu'on retrouve les classes Laser, Batterie et leur classe mère Equipement. Ce sont les equipements dont disposera le robot durant l'exploration de Mars.

Le deuxième est a été la gestion de fichiers, qui contient les classes abstraites permettant de traiter les fichiers de configuration du jeu afin de l'initialiser en fonction de leurs données. Comme par exemple les lasers et batteries disponibles ou bien l'ensemble des paramètres de configuation du robot. Le package interface contient les vues JavaFX qui permettent à l'utilisateur de visualiser l'ensemble des données du jeu, et son état final.

Vous pourrez trouver une page d'accueil, disposant d'une page d'édition donnant des informations sur le jeu, et d'un bouton lancant le jeu, permettant de consulter les résultats du jeu. On retrouve ensuite un package elementsCarte contenant les différents éléments qui composent la zone à explorer. Pour finir, nous avons le package partie où on retrouve les éléments du jeu, tels que la classe Carte, la classe Robot, ou encore la classe Jeu. Notre algorithme applique une stratégie simple pour choisir sa direction, il regarde les 4 cases autour de lui (Nord, Sud, Est et Ouest) puis il choisit la case avec le meilleur ratio d'abord sur la valeur, puis sur la dureté si tous les minerais ont un ratio nul. Donc, si toutes les cases ne possèdent aucune valeur, notre robot choisira le minerai le plus léger. Tant que le robot a assez de batterie pour rentrer à la base, nous continuons de miner avec la même stratégie. Une fois de retour à la base, nous achetons le meilleur équipement possible avec notre score actuel. Pour choisir quelle est le meilleur équipement, nous les classons en fonction d'un ratio du prix sur leur capacité.

Plus la partie avance plus nous minons, et un nouveau cas apparaît si nous le robot a miné toutes les cases autour de lui. Dans ce cas, le robot se dirige vers la case vide la plus éloignée dans le but d'explorer une nouvelle partie de la carte. La partie prend fin lorsque le temps est écoulé, c'est-à-dire quand la NASA nous repère. Nous avons réalisé un autre package avec tous les tests sur notre projet, cela permet de vérifier que toutes nos fonctions renvoient la bonne information, et effectuent correctement ce que l'on souhaite. Pour la répartition des rôles, comme énoncé plus haut nous avons fait plusieurs réunions où nous avons dévelloppé ensemble, cependant la structure du code et les test ont majoritairement été réalisé par Clément. Ce projet n'a pas été simple, nous avons rencontré plusieurs problèmes. Premièrement, nous avons réalisé les tests à la fin. Il aurait été plus efficace et moins fastidieux de les réaliser tout au long du projet. Aussi, le fait de ne pas réaliser plusieurs branches sur le git à été une réelle erreur, car nous avons eu plusieurs problèmes de conflit lors de push sur le git, ce qui nous a fait perdre pas mal de temps. Pour finir nous avions mal évalué la complexité de la stratégie adéquate pour notre algorithme de jeu, et de tous les paramètres qu'il faut vérifier au cours du jeu. Nous n'avions pas assez refléchi au cas où le robot est entouré de cases vides. En conclusion, nous pensons que nous aurions du accorder plus de temps à la refléxion de la stratégie, et à son implémentaion algorithmique. Nous avons opté pour une stratégie simple afin de commencer rapidement son implémentation, mais cela nous a couté du temps au final, que l'on aurait du accorder à la conception de l'algorithme.