RAPPORT TAKENOKO EQUIPE C

Jérémi FERRE - Hugo FRANCOIS - Laura LOPEZ - Prune PILLONE

Chapitre 1

Synthèse

1.1 Fonctionnalités

Toutes les règles de Takenoko ont été implémentées, c'est-à-dire les conditions sur les poses de parcelles et d'irrigations, les déplacements du Panda et du Jardinier et leurs actions respectives. La météo et les aménagements ont été gérés, ainsi que toutes les décisions que le bot doit prendre en conséquence (météo " ?" par exemple). Deux actions par tour sont permises pour les joueurs.

En ce qui concerne la fin de partie, elle est déterminée par la validation du bon nombre d'objectifs, et le joueur ayant fini en premier reçoit la visite de l'empereur. Les spécificités telles que la pousse de bambous sur les parcelles adjacentes à celle où le jardinier finit son déplacement, l'action combinée de la météo "Pluie" et de l'aménagement "Engrais" pour la pousse de bambous, et la gestion des égalités grâce aux objectifs panda ont également été implémentées, tout comme le dernier tour pour les autres joueurs dès que l'un a rempli le bon nombre d'objectifs.

Nous avons aussi ajouté des options de compilation pour les points d'entrées du programme, telles que la définition des stratégies adoptées par les bots, un visualisateur de l'état du jeu ou encore l'activation d'un easter egg (cf README.md).

1.2 Déroulement d'une partie

Le jeu commence par créer les différents joueurs en fonction des arguments passés au programme. Après cela, une partie est créée avec ces joueurs, puis s'ensuit la boucle des tours. Cette boucle va contrôler si un joueur a rempli la condition d'arrêt du jeu. Cette condition vérifie que le joueur a un nombre suffisant d'objectifs accomplis. Ce nombre est calculé en soustrayant le nombre de joueurs à 11.

Après la vérification, chaque joueur effectue un tour qui se déroule de la façon suivante :

— Le dé météo est lancé, le joueur effectue donc l'action pour la météo obtenue (sauf pour le premier tour)

— Le joueur effectue ses actions en fonction du nombre d'actions disponibles, et les choisit selon sa stratégie.

Lorsqu'un joueur accomplit le bon nombre d'objectifs, celui-ci reçoit deux points et la carte objectif empereur. Un dernier tour va donc commencer en partant du joueur qui a reçu la carte empereur, jusqu'au joueur avant lui.

En cas d'égalité du score, les joueurs seront départagés par le cumul des points des objectifs panda accomplis.

1.3 Tests et confiance

L'affichage du nombre de parties sans fin sur une série de n parties est un important indice de fiabilité, nous l'avons mis en place afin de pouvoir montrer la confiance que nous avons en notre code. Nous affichons aussi la durée du jeu, le nombre de tours, les stratégies utilisées et leur efficacité ainsi que le nombre d'égalités. Nous avons produit un maximum de statistiques afin d'être certains de la fiabilité de notre code. Aujourd'hui, nous avons confiance en notre projet, grâce aux statistiques et aux nombreux tests. A l'exécution de 10 000 parties, 99,98% des parties lancées se terminent correctement.

En ce qui concerne les tests, nous avons suivi la structure suivante :

— Cas certains de fonctionnement (ex : Ajout d'un bambou à l'inventaire du joueur lors du déplacement du panda sur une parcelle avec des bambous sans enclos)
— Cas certains de non fonctionnement (ex : Pas de pousse de bambou lors du déplacement du jardinier sur une parcelle non irriguée)
— Cas limites (ex : Tenter de piocher une parcelle dans une pioche vide)

Pour notre dernière release, la huitième, nous avons prévu et réalisé beaucoup de tests, car nous avions des tâches consacrées au debug et ces tests nous permettaient d'avancer pas à pas pour trouver les possibles cas d'erreur.

1.4 Architecture

Au niveau de l'architecture, nous avons choisi d'implémenter plusieurs design patterns :
— Strategy
— Decorator
— MVC

Nous avons utilisé le polymorphisme au maximum dans les cas qui nous paraissaient les plus pertinents.

Les bots que nous avons programmés sont représentés par un objet Bot commun, différenciés par la stratégie qu'ils utilisent pour choisir leurs prochaines actions. Le pattern strategy est très utile pour ne pas redéfinir plusieurs bots et externaliser les comportements de prise de décisions.

Pour les parcelles, le pattern decorator a été utilisé pour avoir une implémentation et une utilisation plus simple des aménagements. Lorsque le décorateur a été mis en place pour la première fois, le reste fut très rapide.

Nous avons également tenté de respecter le pattern MVC au maximum, de par la grande séparation entre l'implémentation des bots et la totalité du moteur de jeu. Il a donc été plutôt simple de créer une interface graphique de visualisation pour simplifier le debug.

1.5 Pistes de réflexion

Après avoir demandé conseil à l'un de nos anciens professeurs, nous nous sommes penchés sur le machine learning. Nous avons tout d'abord essayé de simuler un neural network mono-couche puis multi-couche. Cependant, nous nous sommes rapidement confrontés à un gros problème

du neural network : les sets d'entraînements et les inputs/outputs. Nos recherches se sont aussi portées sur le deep learning avec la méthode Q-Learning. Nous nous sommes aussi renseignés sur la recherche dite Monte-Carlo, mais faute de temps et de par la complexité de cette méthode, nous n'avons pas poussé nos recherches plus loin.

Chapitre 2

Avis sur notre projet

2.1 En quoi notre projet est-il un bon projet ?

2.1.1 Planification

Nous nous sommes organisés avec 8 releases et du découpage vertical, c'est à dire que chaque partie du projet est améliorée lors de la nouvelle release (moteur de jeu, bot et logger). Par exemple, en ce qui concerne les déplacements du panda et du jardinier : lors de la release 2, ils ne pouvaient se déplacer que sur une parcelle adjacente, puis dans la release 3 en ligne droite sur tout le plateau, dans la 5, le jardinier faisait pousser les bambous sur les parcelles irriguées de même couleur, etc.

Certains concepts ont été introduits plus tard dans le projet tels que la météo, les irrigations ou les aménagements. Nous avons donc découpé nos versions avec l'idée d'implémenter un nouvel élément de jeu à chaque release.

2.1.2 Organisation

Personne dans le groupe n'aura effectué une tâche qui lui été attribuée à la release précédente. Cela nous a permis de bien connaître le code du projet dans sa globalité, ainsi que de créer un roulement entre les tâches faciles et difficiles (notamment le bot).

2.2 En quoi notre projet est-il un mauvais projet ?

Le principal point négatif est la durée de nos parties. En effet, 1000 parties à 4 joueurs, chacun possédant une stratégie différente, durent presque 10min. Nous estimons que cette durée est trop grande.

2.2.1 Code de mauvaise qualité

Un exemple de code de mauvaise qualité serait la méthode getWeatherMove dans la classe Bot. Cette méthode renvoie l'action météo que réalisera le Bot. Cette méthode est composée uniquement d'un switch couvrant seulement des cas basiques et ne prend pas en compte la

Stratégie adaptée par le Bot, et pourrait donc être grandement améliorée, tant que sur les fonctionnalités que sur le code en lui même.

Chapitre 3

Rétrospective

3.1 Points à améliorer

Notre plus grand défaut durant ce projet était de ne pas faire les tests directement, nous avons donc pris un peu de retard sur ce point et avons du retarder une des releases pour finir les tests de la précédente.

Nous avons également eu quelques difficultés pour la rigueur de nos documentations de classes et de méthodes, surtout lors des premières releases. Il a donc été nécessaire de rattraper ce manque d'indications lors des semaines suivantes.

Un point qui nous a aussi fait défaut sur ce projet est la mise en ligne des tickets. Nous avions tendance à les ajouter release par release et peut être à manquer d'anticipation. Nous avons changé cette façon de faire à partir de la release 6.

3.2 Points à conserver

Avoir un roulement entre les différentes tâches a été très profitable. Cela nous a permis à tous de bien comprendre le code des autres, travailler diverses compétences et ne pas rester bloqués sur une tâche.

La communication de notre équipe a vraiment permis une bonne motivation, même lors des tâches difficiles. Lors d'un blocage sur une implémentation ou un choix de conception à réaliser, une mise en commun sur le Slack était faite directement, afin de tous donner notre avis, ce qui a débloqué plusieurs situations délicates (par exemple, pour la manière d'implémenter les aménagements ou la météo. Des réunions étaient également organisées chaque semaine, ce qui permettait de discuter de conception et de se mettre à jour sur les tâches futures et l'avancement du projet.

Nous avons également su exploiter les compétences et préférences de chacun et c'est un point très important, que nous allons essayer de conserver lors de nos prochains projets. L'organisation acquise au cours de ce semestre s'est concrétisée par une collaboration efficace.