PS5
TAKENOKO

LES MATELOTS DES 7 MERS

ARCIL Alexandre
CLODONG Yann
COGNE Gabriel
D'ANDREA Wiliam

Université Nice Sophia Antipolis | POLYTECH NICE-SOPHIA

projet2-ps5-20-21-takenoko-2021-les-matelots-des-7-mers

1 - Introduction

“ Le but du jeu Takenoko est de s'occuper d'un panda, cadeau de l'empereur de Chine à celui du Japon, tout en prenant soin de la bambouseraie impériale, entretenue par un jardinier. Les joueurs cultivent des parcelles de terrain, les irriguent, et y font pousser l’une des trois variétés de bambou (vert, jaune ou rose) par l’intermédiaire de ce jardinier. ”

Wikipedia

2 - Synthèse

Features présente dans la version FINAL (V1.0) du projet PS5

Le système de parcelles au format hexagonal

- Il y a un nombre limité de parcelles de chaque couleurs
- Pour placer une parcelle, il faut que la tuile soit adjacente à au moins deux autres
- Pour placer une parcelle, il faut en piocher 3 puis en choisir une

Les irrigations :

- Les irrigations doivent être piochés avant d’être utilisées
- Une irrigation doit être relié à un réseau de parcelles irriguées
- Une irrigation est situé uniquement entre 2 parcelles

Les Objectifs :

- Les objectifs ne peuvent être tirés qu’une seule fois par tour (sauf si le tour est régi par une condition climatique vent)
- Objectifs parcelles :
    - Sont de 3 formes différentes : triangles, ligne, C, losange
    - Possèdent des couleurs de parcelles différentes : vert, jaune, rose
    - Ont des scores différents
- Objectifs pandas :
    - Ont des quantités de bambou à manger par le panda différentes
    - Ont des scores différents
- Objectifs jardiniers :
    - Ont des objectifs de hauteur de bambou différentes
    - Ont différentes couleurs de bambou et parcelles : vert, jaune, rose
    - Ont des aménagements particuliers
    - Ont des scores différents
- Empereur

Panda :

- Se déplace en ligne droite uniquement
- Mange des bambous de toutes couleurs

Jardinier

- Se déplace en ligne droite uniquement
- Fait pousser les bambous autour de lui si les tuiles sont irrigués uniquement

Aménagements

- Engrais - au lieu que un seul bambou pousse, on en a 2
- Enclos - le panda ne peux pas manger de bambou dessus
- Étang - on peut faire pousser une parcelle même si elle n'est pas irrigué

Météo

- Soleil : un joueur peut faire 3 actions différentes les unes des autres
- Pluie : un joueur peut ajouter une section de bambou n'importe ou (si la parcelle est irriguée)
- Vent : un joueur peut faire 2 actions identiques
- Orage : un joueur peut faire bouger le panda n'importe ou, il n'y a aucune contrainte de ligne
- Nuages : Le joueur peut tirer un aménagement et le placer (ou le stocker pour plus tard) - Notre bot stocke juste l'aménagement
- ? : Le joueur peut choisir la météo de son choix

5 bot qui jouent avec plusieurs stratégies différentes :

- Bot Rush Parcelle : ce bot ne vas essayer d'effectuer que des actions pour valider des objectifs parcelles : ce bot ne peux plus jouer lorsqu'il y a le nombre maximal de parcelle sur le board, il abandonne donc
- Bot Rush Panda : ce bot ne vas essayer d'effectuer que des actions pour valider des objectifs panda : lorsque le bot n'a plus d'objectifs panda, il essaye de prendre des objectifs parcelles et jardinier et essayer de les résoudres aléatoirement, si le bot passe trop de tour a rien faire ou faire la même chose, il abandonne
- Bot intelligent 1 : il essaye de résoudre les objectifs parcelle et jardinier dans un ordre quelconque. Le bot abandonne lorsqu'il ne peut plus faire d'actions.
- Bot intelligent 2 : va essayer de réaliser tout types d'actions afin de réaliser des objectifs différents.

Chaque bots, au bout d'un moment, peut, potentiellement ne plus pouvoir faire d'action, dans ce cas, selon les bots, soit il passera son tour, soit il essayera de faire une action au hasard. Si deux bots ne peuvent plus faire d'actions, la partie s'annule, autrement dit, lorsque les deux joueurs abandonnent, la partie est annulée, c'est pour cela qu'il peut arriver, selon les bots qui jouent, que plus ou moins de parties soient annulées.

La Victoire se fait sur le nombre total de point gagné en accomplissant des objectifs puis en cas d'égalité sur le nombre de point gagné avec des objectifs panda

Features manquantes dans la version FINAL (V1.0) du projet PS5

- Faire débuter les bots avec 3 objectifs dans leur inventaire
- Créer un bot rush gardener, nous avons créé des méthodes intelligentes afin de réaliser cet objectif mais nous n'avons pas eu le temps d'en créer un.

3 - Autocritique

A - Respect des concepts de POO

SOLID

Single responsibility principle

Mis à part la Classe board ou parcel, les responsabilités ont été distribué en classe tel que la vérification et complétion d'objectifs qui ont été implémentés dans les classes d'objectifs.

Open/Close principle

Nous avons respecté le principe car toutes les classes sont fermées aux modifications, mais elles sont toujours substituables par les éventuelles classes enfants.

Liskov substitution principle

Ce principe est violé dans l'objectif panda, car pour connaître le nombre de bambou mangé par le panda par l'action de tel ou tel joueur il faut connaître qui est ce joueur. Or, pour vérifier si le joueur a fait manger assez de bambou au panda, la carte objectif a besoin de connaître la carte de statut du joueur, ce qui nous a incité à utiliser une construction de l'objet en deux temps : l'une pour ses paramètres propres, et l'autre pour le jeu de données à vérifier au moment du tirage de la carte.

Interface segregation principle

Nous n'avons pas utilisé d'interface dans notre projet, donc nous n'avons pas appliqué ce principe. Nous aurions pu cependant en créer une interface pour "Parcel".

Dependency inversion principle

Nous avons globalement pu respecter ce principe, par exemple, une partie fait jouer des bots en utilisant la classe abstraite “Bot” ainsi que la méthode “playTurn”. De plus, dans les bots, nous avons utilisé plusieurs méthodes que nous avons override

GRASP

Expert

Nous avons essayé d’appliquer ce principe notamment pour les bots, en créant des méthodes que nous avons @Override afin d’adapter des méthodes génériques aux comportements de chaque bot voulu.

Créateur

La création des cartes objectifs, parcels suivent ce pattern, en effet, ces structures ne sont instanciées que dans leurs decks respectifs, et nulle part ailleurs

Faible couplage

Le couplage a été minimisé par la centralisation de la plupart des fonctions principales sur le Board, ainsi la plupart de ces fonctions sont exécutées en passant par 2 ou 3 instances.

Forte cohésion

Même si la cohésion en plutôt faible dans la classe Parcel, ce pattern est en revanche utilisée dans diverses autres classes telles que le Jardinier, Panda, Objectifs etc

Bon points

Respect des deadlines hebdomadaire

En dehors de la semaine de projet, la plupart des issues étaient closes à la fin de la semaine, celles qui ne l'étaient pas n’ont pas ralenti la suite et ont seulement été ajoutées à la charge de travail de la semaine suivante.

Révision du découpage initial

Le découpage initial avait pas mal de défaut mais nous nous en sommes aperçu avant qu’ils ne surviennent ce qui nous a permis de le rectifier en souplesse.

Mauvais points

Découpage initial

Le projet a mis un peu de temps à se lancer à cause d’un découpage un peu long, et moyennement réaliste. En effet, nous avons découpé selon un découpage métier, et non un découpage horizontal. Au début, nous voulions d’abord ajouter directement des éléments de jeu mais sans ajouter des bots, même random. Nous avons donc slicé à nouveau le projet afin de pouvoir fournir de la valeur à chaque étape.

Organisation pendant la semaine de projet complète

Au début du projet, nous étions bien organisés et nous avons bien réussi à utiliser les outils sur git (les issues, milestones...). Cela nous était très utile du fait que l'on n'était qu'une séance par semaine ensemble, et donc, les issues étaient très pratiques car on savait exactement ce qu'on avait à faire. Cependant, quand est arrivé la semaine tous ensemble, nous avons moins pris le temps de créer chaque fois des nouvelles issues et milestones (l'idéal aurait été une par jour). Nous étions toujours ensemble et faisons globalement des rétrospectives ensemble à chaque matinée, donc on savait vraiment sur quoi avancer sans forcément en créer des issues. Cependant, nous nous rendons compte maintenant qu'il aurait été beaucoup mieux de chaque soir, analyser ce qu'on avait fait, et de prévoir les issues du lendemain, afin de ne pas perdre de temps inutilement.
D'un point de vue 'valeur apportée', même si toutes nos tâches étaient bien situées dans nos milestones, nous avons mis des fois 2-3 jours pour valider une seule milestone, avec le recul, on se rend compte qu'on aurait mieux fait de séparer les milestones afin qu'une personne extérieure puisse suivre réellement l'avancée du projet car de l'extérieur, on n'aurait pas forcément pu voir la valeur apportée.

Tests

A la fin du projet, nous avons également été pris par le temps, en effet, pour de nombreux algorithmes, nous avons essayé au maximum de faire des méthodes grâce à la méthode Test Driven Development, ce qui nous a pris beaucoup de temps, et à la fin, nous avons eu du mal à finir le deuxième bot intelligent (FourthBot), et nous n'avons donc pas réussi à faire tous les tests que nous voulions. Pour les bots nous aurions aimé pouvoir faire plus de tests, car souvent, lorsque l'on ajoute une nouvelle fonctionnalité au jeu, certains tests ne fonctionnaient plus, on devait donc souvent reprendre le code.

4 - Rétrospectives

Au début du projet, nous avions du mal avec les règles car celles-ci pouvaient être assez compliquées à première vue, ce qui nous a, souvent, mis en erreur. Pour les prochains projets, il nous faudra prendre plus de temps pour comprendre les règles, avant de se lancer aveuglément dans un découpage qui pourrait ne pas être complet ou ne pas être adapté.

Au niveau de l'organisation, il nous faudra être plus rigoureux. Il nous faudra, plus ou moins tous les soirs (par exemple), prendre le temps de réécrire les Milestones et issues, afin de pouvoir voir l'évolution dans le temps. il nous faudra aussi prendre la rigueur de chaque fois, bien affecter un commit à une issue, car sur certaines issues, nous avons oublié de le faire.

Au niveau du code en lui-même, nous avons pu avoir des difficultés notamment avec la mise à jour des bots avec les nouvelles fonctionnalités qui arrivaient au fur et à mesure. Même si nous avons réussi à le faire un peu, il nous faudra par la suite créer des méthodes encore plus génériques qui pourront s'adapter aux nouvelles évolutions, ce qui nous permettra d'avoir des tests plus génériques mais aussi de ne pas perdre de temps.