GRP2 – Aurélien Romain
Pierre Lucie Nathan Clément

Rendu Ecrit

Projet tutoré Symbioz

Groupe D

Sommaire

I) Introduction

Description du sujet

À la tête d'une équipe de Kroguls et de Crapit, et en essayant d'implanter le plus de Zerbs possible à sa couleur, chaque joueur cherche à étendre son influence sur diverses zones du plateau. Ce dernier est constitué de 3 disques concentriques, respectivement découpés en 4, 8 et 12 zones.
Chaque joueur commence dans le quart de plateau situé devant lui et y dispose un certain nombre de Zerbs, des petites plantes dont sont friands les Crapits, eux-mêmes les mets favoris des Kroguls carnivores.
Un tour de jeu se déroule en 4 phases :
. introduction des Zerbs, Crapits ou Kroguls : chacun a 10 points d'action à dépenser pour placer ses pions. 1 point par Zerb, 3 par Crapit, 5 par Krogul. Bien évidemment, chaque placement est régi par des règles bien précises ;
. multiplication des Zerbs : chaque groupe de 4 Zerbs dans une même zone (ou de 3 Zerbs en zone centrale, plus fertile) engendre une nouvelle Zerb "gratuite" dans la même zone ou dans une zone directement adjacente ;
. reproduction des Kroguls : un Krogul se clone une fois par tour (dans la même zone ou dans une zone directement adjacente), dans la mesure où il mange un Crapit (adverse ou ami) situé dans sa zone ;
. reproduction des Crapits : un Crapit pour survivre doit pouvoir manger une Zerb (adverse ou amie) ; de plus, deux Crapits de même couleur donnent naissance à un Crapit dans la même zone ou dans une zone directement adjacente.
À chaque fin de tour, si une zone contient 12 Zerbs (et est donc remplie), le propriétaire des Zerbs réalise une "Symbioz".

Le premier joueur à réaliser 3 "Symbioz" est le vainqueur. Au bout de 12 tours de jeu, s'il n'y a toujours pas vainqueur, c'est celui qui en a réalisé le plus qui gagne.

Fonctionnalités demandées

Les fonctionnalités attendues au début du projet avec MoSCoW:

-> Plateau de jeu Symbioz (s)
-> Javadoc (s)
-> Junit et tests (s)
-> Unités (m)
-> Tours sans phases (m)
-> Tours avec phase 1 (m)
-> Tours avec phase 2 (m)
-> Tours avec phase 3 (s)
-> Tours avec phase 4 (s)
-> Victoire détectée (m)
-> Jolie interface (c)
-> Jeu console (m)
-> Reproduction des Zherb (m)
-> La création des zones. (Cercle extérieur, milieu, zone centrale) (m)
-> Des couleurs pour chaque joueur. (s)

Les fonctionnalités ajoutées au projet par le client entre le sprint 1 et 2 avec MoSCoW :

-> La règle maison, nous voulions faire une tornade pour chaque joueur placée au départ sur la partie fertile du plateau, le déplacement de cette tornade couterait 4 en déplacement et s'activerait à la fin de la phase 1 pour détruire 5 items sur cette case comme des Zherb, des Krogul et des Crapits. (c)
-> Pas plus de 2 couleurs de Zherb sur une case (s)
-> La cagnotte = un joueur peut garder la moitié de ce qu'il n'a pas dépensé pour le prochain tour (c)
-> Un bot aléatoire qui peut à toute phase jouer au hasard de manière uniforme parmi toutes les actions possibles. API joueur (m)

Projet livré

-> Les tours sans phases ont été réalisés. (m)

-> Un observer a été créé : il permet de mettre à jour l'interface graphique si le plateau change. (s)

-> La phase 1 : on y demande à chacun des joueurs le placement des unités. L'interaction se fait par la console en demandant aux joueurs des choix selon une intervalle donnée. L'interaction ne se fait pas directement avec l'interface graphique (mouseListener) car le système d'intervalles permet de facilement demander à un bot de faire des choix (L'interaction des joueurs physiques et bots est du même type). (m)

-> Tests : Chaque création d'objets et le bon fonctionnement de leurs méthodes a été vérifié au fur et à mesure de leurs programmations pour que chacun puisse utiliser les méthodes des autres sans problème. Junit avait commencé à être mis en place mais par manque de temps nous n'avons pas pu faire de tests avec. (s)

-> L'implémentation de la victoire est faite. (m)

-> Une reproduction des Zherb aléatoires sur les cases adjacentes est réalisée. (m)

-> Création de la javadoc dans le dossier /Projet/docs est faite. (s)

-> Les joueurs peuvent choisir leurs actions.

-> Un bot, c'est un joueur non contrôlable et qui fait des choix au hasards en fonction des questions posées, il a les mêmes interactions qu'un joueur normal, avec un peu plus de temps nous pourrions l'améliorer et le rendre plus intelligent en biaisant son aléatoire pour qu'il évite de finir la phase sans dépenser de mana.

-> Les tours : c'est ce qui permet d'initialiser le jeu, les phases, les joueurs, le plateau. Les différentes phases y sont appelées et les joueurs y jouent les uns à la suite des autres ce qui permet d'orchestrer le jeu. Ce qu'il manque c'est de finir le jeu s'il y a une victoire des joueurs, ça n'a pas pu être mis en place parce qu'on n'a pas fait toutes les phases même si la détection de victoire fonctionne (il manque la phase 3 et 4). (s) Avec un peu plus de temps nous aurions pu coder la phase 3 et 4 qui ressemblent beaucoup à la phase 2.

-> La phase 2 a été achevée ainsi que la phase 1. (m)

-> La règle de pas plus de 2 couleurs de Zherb sur chaque case fonctionne. (s)

-> La cagnotte fonctionne et donne la moitié des manas non dépensés arrondi à l'entier inférieur. (c)

-> La création des zones est faite. (Cercle extérieur, milieu, zone centrale) (m)

-> Chaque case possède une liste de tous ses voisins, créé à l'initialisation. (m)

-> Une interface graphique représentant le plateau de jeu et qui montre le placement de chaque Zherb, Krogul et Crapits. (s)

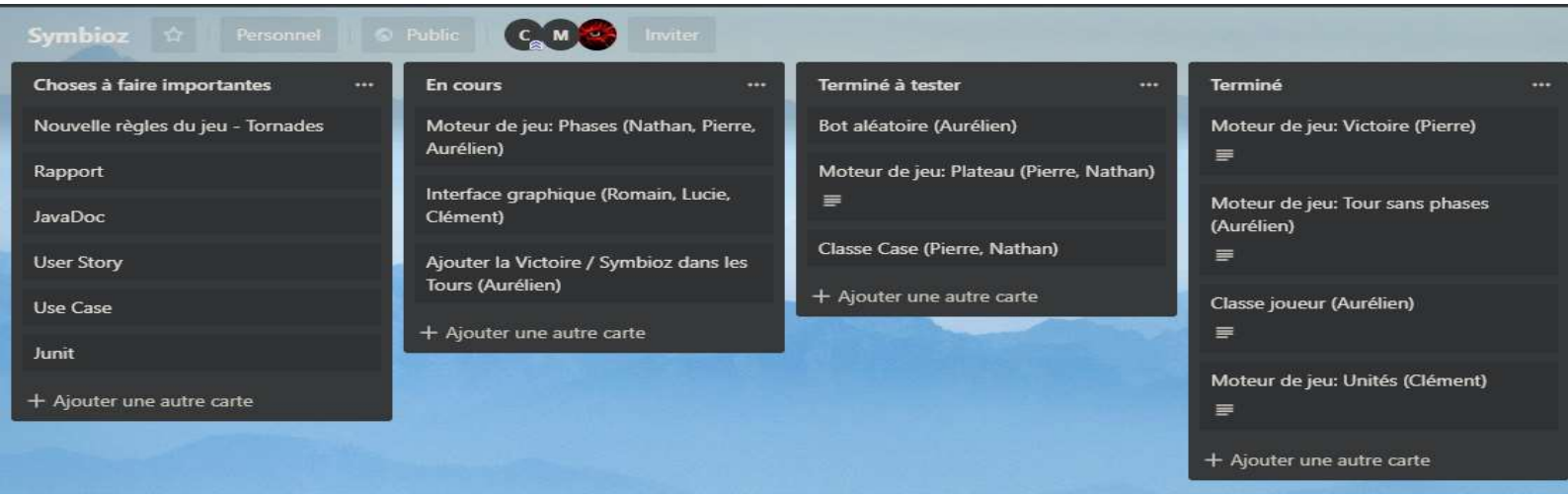
-> Des couleurs pour chaque joueur est réalisée. (s)

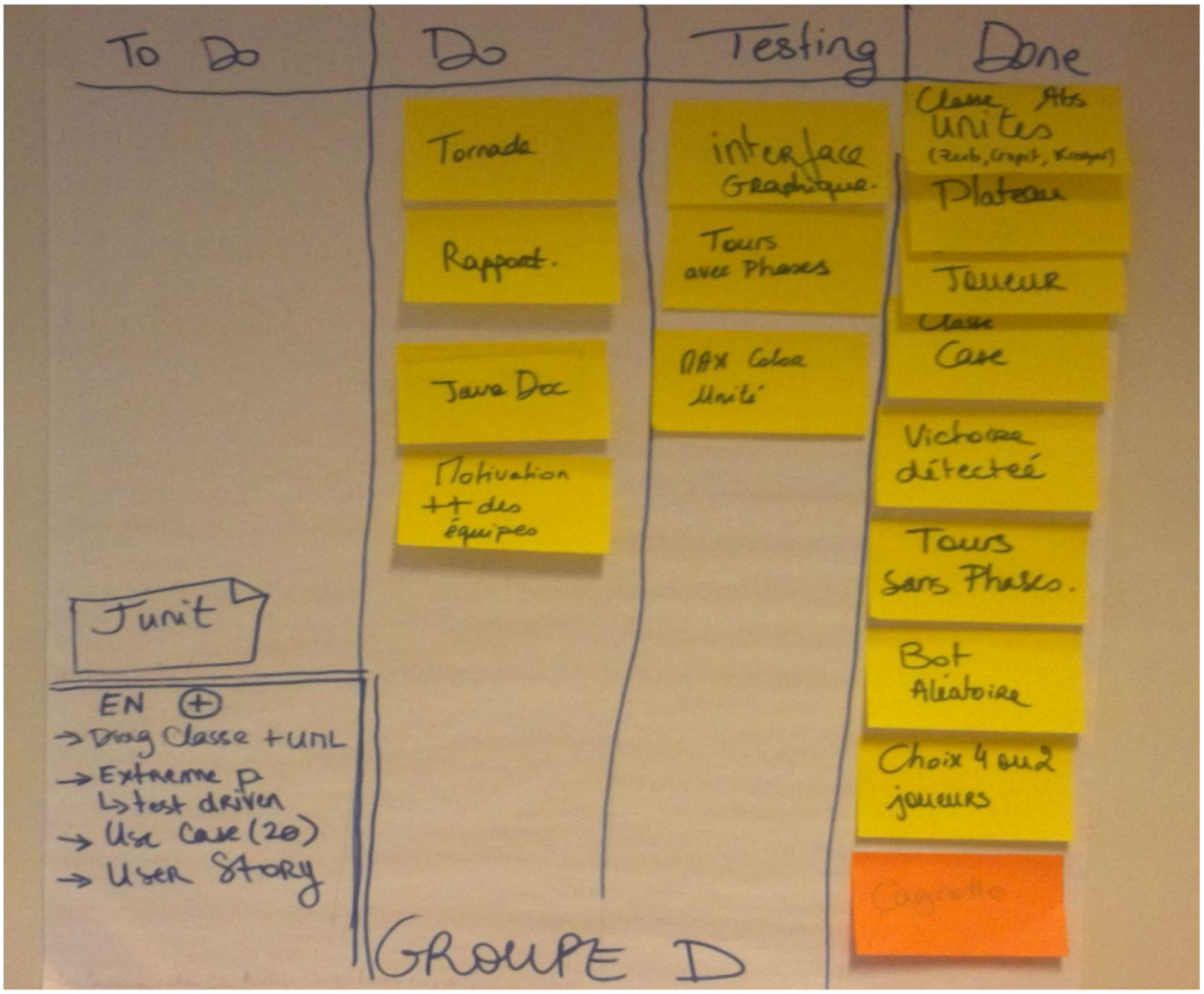
-> Les couleurs des Zherb, Krogul et Crapits sont différenciées.

II) Méthodologie de l'équipe

Méthodologie de l'équipe et répartition des tâches

Nous avons utilisés un Scrum sprint board en ligne https://trello.com/b/nPKyd1Yp/symbioz mais aussi physique en salles de TP (le plus à jour est la version physique).





Nous avons aussi beaucoup communiqué par discord.

Sprint retrospective

Sprint 0 : Initialisation du projet. Mise en place de la méthode scrum, création du serveur de communication discord, du trello et du git.

Sprint 1 : Explication de la mise en œuvre :
On créé d'abord un objet Plateau. Puis la mise en place du jeu se fait en fonction des Tour.
Il y a différentes Phases dans un Tour. Lorsqu'une phase est finie on utilise un booléen qui sera mis à TRUE et on passe à la phase suivante (comme une boucle while(1) on attend les actions de l'utilisateur).
Chaque Phase a des interactions différentes. On va interagir avec le Plateau. On mettra par exemple des objets Unités dans des Cases du Plateau selon la Phase.
A la fin on détectera la fin de partie (soit une Symbioz: victoire, ou bien si on a fait 12 tours).

Lors de ce sprint : Création de la base de l'interface graphique. Création des packages Unités, Joueur, Tours, Interface.

Entre le sprint 1 et 2 : Rencontre avec le client.
Demande de nouvelles fonctionnalités telles que le plus de 2 couleurs par Zherb (S), Cagnotte, le joueur récupère la moitié des points non dépensés pour le prochain tour (C), le bot aléatoire (M) et la règle maison : ajout d'une tornade

Sprint 2 : Lors de ce sprint : Voir « projet livré ».

Nous aurions pu finir plus de fonctionnalités du projet si nous avions eu un sprint de plus, il nous restait à faire la phase 3 et 4 semblables à la phase 2 donc facilement réalisable, la règle maison manquait aussi ce qui aurait été plus simple à faire après la réalisation de la phase 3 et 4.
Nous aurions dû dès le départ nous focaliser sur la jouabilité, en effet, sans la phase 3 et 4, le jeu est moins intéressant.
La méthode MoSCow et Scrum nous a pris beaucoup de temps (organisation de l'équipe et division du projet en taches) ainsi que la réalisation de la phase 2, ce qui nous a retardé pour la suite.
Nous avons compris que les tests sont importants et le fait de ne pas avoir implémenter de Junit est un problème, nous aurions dû l'intégrer au fur et à mesure de l'écriture du code comme nous l'avons fait pour la Javadoc. Il ne restait ensuite qu'une jolie interface à faire avec laquelle nous aurions pu jouer au lieu de jouer sur la console.
Il aurait été aussi intéressant de pouvoir créer d'autres types de bot, comme un bot fonctionnant avec l'algorithme de Monte-Carlo ou bien MinMax. Grâce à notre API Joueur il serait facile d'intégrer de nouveaux bots pour qu'ils jouent entre eux. Voir, avoir plusieurs types de bots disputant un match de Symbioz et connaître quel est le meilleur algorithme pour résoudre un tel jeu.

III) Schémas supplémentaires

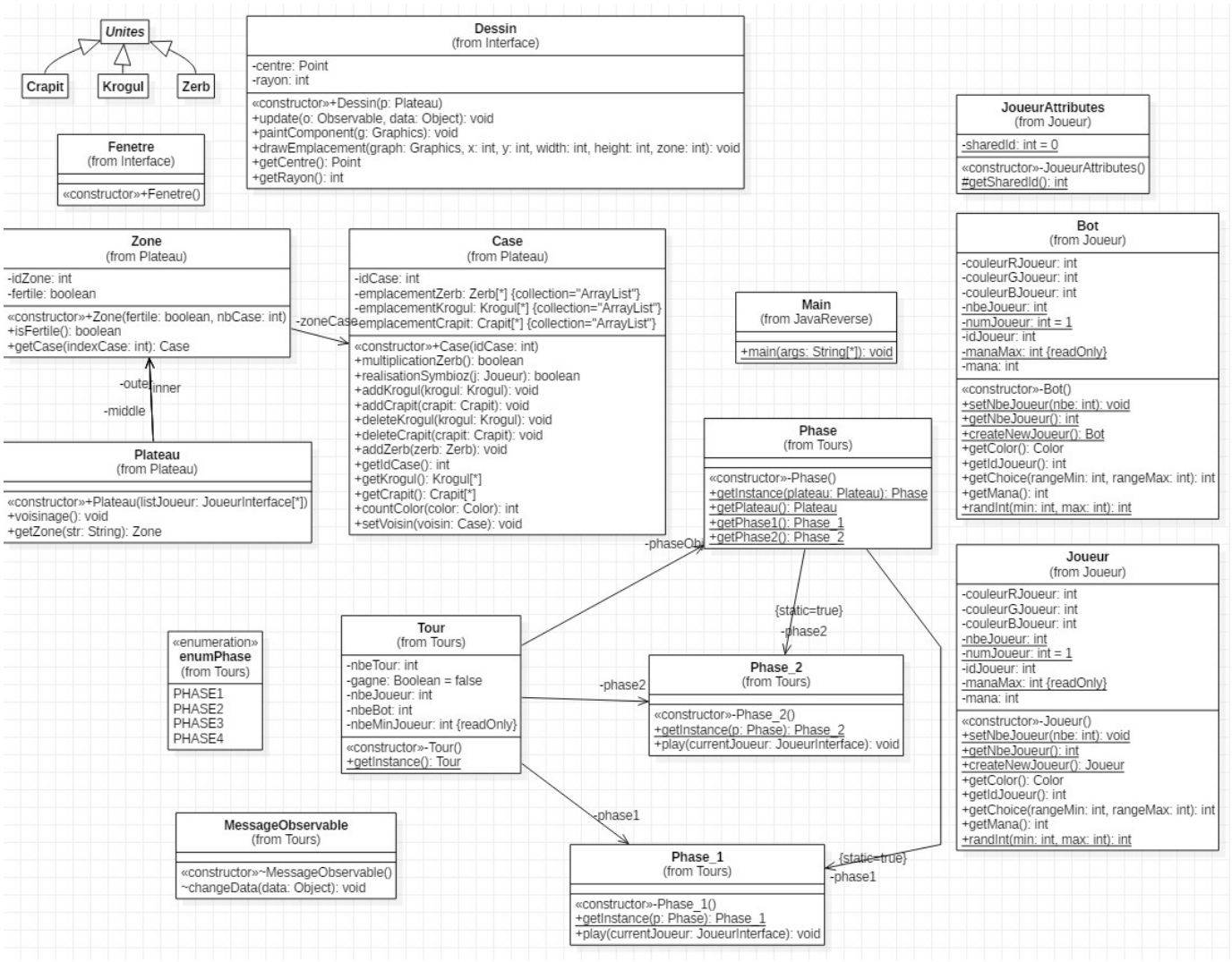


Diagramme de classe