Rapport de conception - Projet S7 : Smallworld

Thierry Gaugry
Benoit Viguier
INSA de Rennes
4INFO, groupe 2.2

12 novembre 2014

Table des matières

1 Présentation du sujet et des rêgles

Ce projet consiste en un jeu de plateau au tour par tour, basé en partie sur le jeu de plateau *Smallworld*, édité par Days of Wonder, et sur la série de jeux vidéo *Civilisation*, développée par Firaxis. Dans ce jeu, chaque joueur déplace ses unités sur la carte et essaie d'avoir plus de points que son adversaire au terme d'un nombre de tours décidé au début de la partie. A la fin du tour de chaque joueur, celui-ci gagne des points en fonction du type de case occupé par ses unités et de leur race.

Très vite, les joueurs chercheront a prendre le contrôle des cases rapportant le plus de points, et il leur est permis de faire s'affronter leur unités pour la domination de celles-ci. Dans ce monde vraiment trop petit, les joueurs combatteront donc becs et ongles pour une position privilégiée sur le plateau. Cela pourra éventuellement entrainer le décès prématuré d'un des joueurs, dans le cas où son armée se retrouverait réduite à l'état de lambeaux et de souvenirs, ce qui offrirait la victoire à son adversaire. Dans notre grande bontée, nous permettons aussi aux joueurs les plus sensibles de déclarer forfait, le monde de Smallworld étant étonnement cruel !

Ce rapport présente l'étude des spécifications, réalisée dans l'optique de développer ce jeu au cours du semestre. On parlera en premier lieu des différentes méthodes de création de partie, puis du déroulement de la dite partie. Nous parlerons ensuite des contrôles que l'utilisateur pourra avoir sur le jeu. Enfin, nous présenterons les interfaces nécessaires et le diagramme de classe qui servira lors de la réalisation.

2 Lancement de la partie

Deux cas distincts peuvent se présenter lors du lancement de la partie :
— La création de partie
— Le chargement de partie

Ces deux cas seront implémentés via le design pattern monteur, comme convenu avec ce qui nous était imposé. On peut en effet trouver des étapes similaires lors de chacun d'entre-eux., telle qu'une étape de création de carte, la création de 2 joueurs, de leurs unités, etc. Nous étudierons donc ces deux cas l'un après l'autre.

2.1 Création de partie

La création de partie utilise les informations passées par le menu principal pour créer une nouvelle partie. Ces informations regrouppent le nom des joueurs, leur peuple et la taille de la carte. La méthode de création est appelée par le monteur de partie, et celle-ci est détaillé sur le diagramme de séquence 1.

2.2 Chargement de partie

Le chargement de sauvegarde permet à un joueur de reprendre une partie en cours. La restauration de partie s'effectue en trois étapes :
— Restauration des joueurs : Les noms de joueurs, leur tribu, leurs points.
— Restauration des unités : Leurs points de vie, propriétaire et leur position.
— Restauration de l'état du jeu : Tour de jeu global et joueur courant.

La récupération de l'état du jeu permet ainsi de reprendre une partie à l'endroit exact où on l'avait laissé. Il est ainsi possible de charger une partie sauvée entre deux mouvements d'unités. Le monteur de partie réalisant le chargement doit faire appel à un système de chargement de sauvegarde ; celui-ci sera détaillé davantage au cours de ce rapport. Les méthodes appelées par le monteur sont détaillées sur le diagramme de séquence 2.

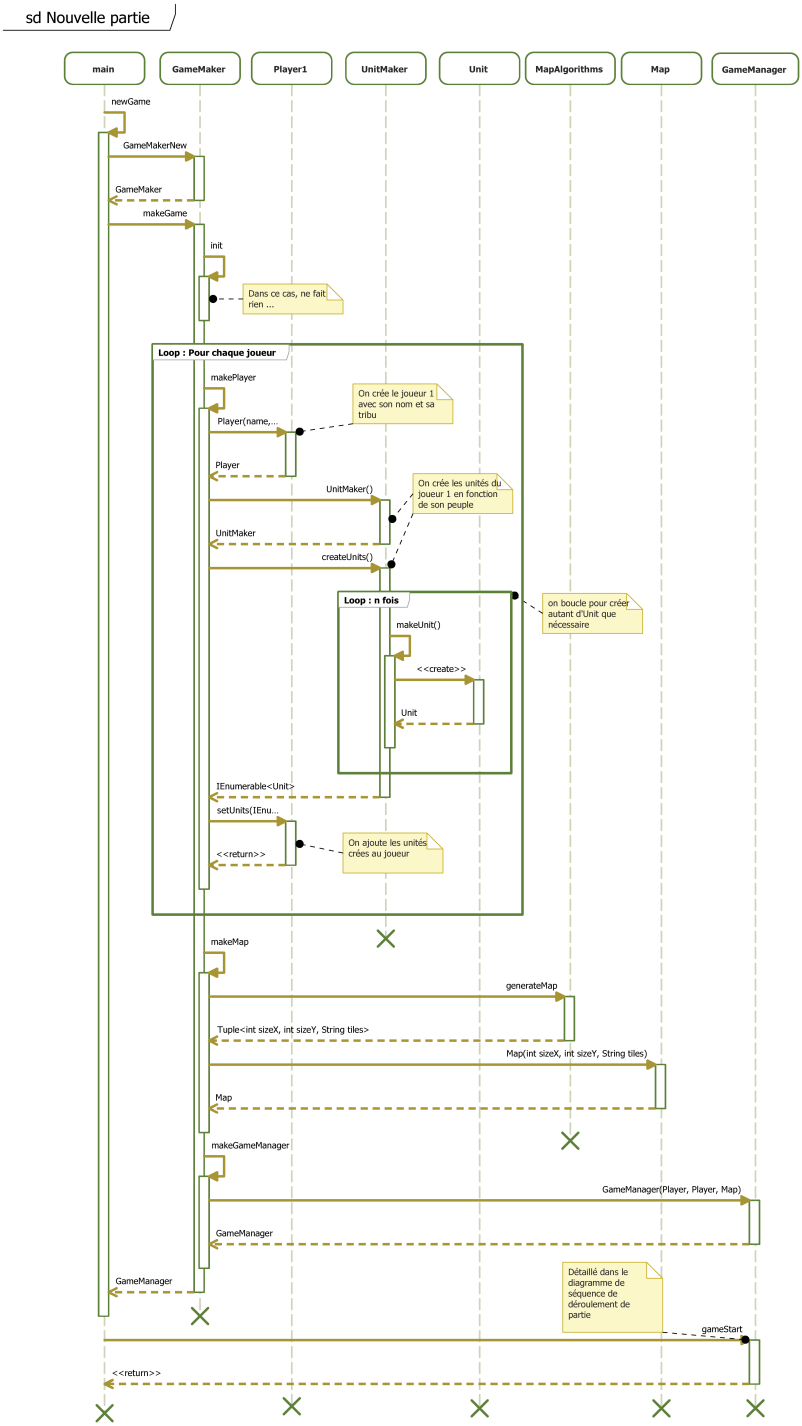

FIGURE 1 – Creation de partie

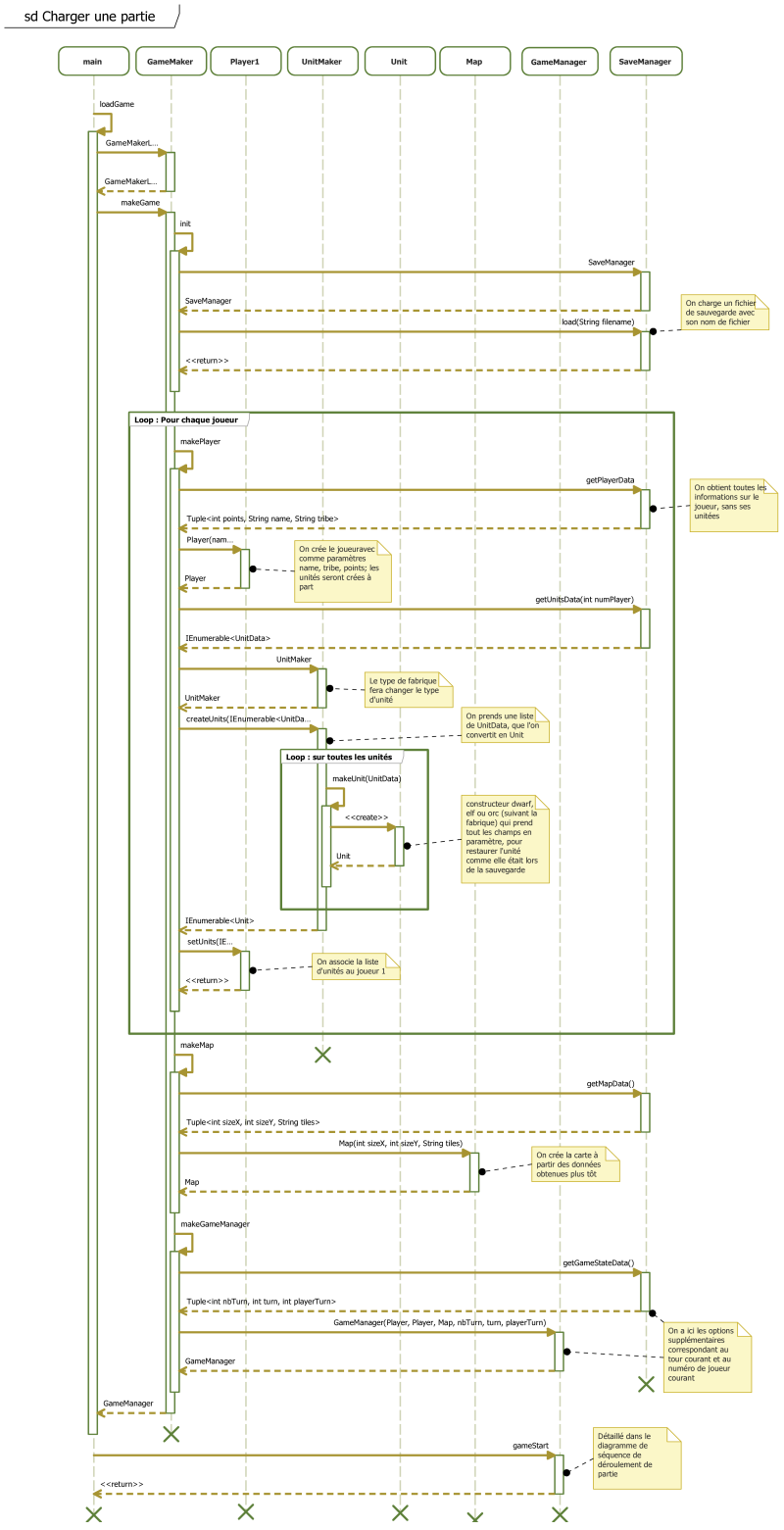

FIGURE 2 – Chargement de partie

3 Deroulement de la partie

3.1 Tours de jeu

Une partie dure jusqu'à ce qu'une des conditions de fin de partie soit remplie. Ces conditions inclues, mais ne sont pas limitées à :

— L'épuisement du compteur de tours.

— Tout les adversaires d'un joueur sont mort (ils ne disposent plus d'unité).

Le nombre de tours est directement implémenté dans le GameManager, et est testé à chaque tour de jeu. L'autre condition est testée après chaque mouvement. On veut ainsi être certain que la partie se termine dès qu'un seul joueur est encore vivant. On peut bien sur imaginer d'autres conditions, telles que la prise d'une case particulière ou la mort d'une certaine unitée. Ces conditions sont implémentables dans l'architecture actuelle car un changement dans l'état du jeu implique qu'il y ait eu un mouvement ou une fin de tour, les deux cas où l'on teste les conditions de fin de partie.
Le déroulement de la partie est classique : les joueurs jouent chacun leur tour, bougent leurs unités tant qu'elles disposent de mouvement et décident quand ils veulent terminer leur tour (possiblement car il ne peuvent plus faire d'actions). Ce déroulement de la partie est détaillé davantage sur le diagramme de séquence 3.

3.2 Mouvements et attaques

La fonctionnalité d'attaque est incluse dans un mouvement. Si une unité se déplace sur une case où se trouve une ou plusieurs unités adverse, la procédure de combat est lancée. De ce fait, l'interface graphique n'aura qu'a implémenter uniquement le contrôle sur le déplacement, et pas sur l'attaque. Les combats sont réalisés séquentiellement eux aussi ; l'attaquant attaque toujours au défenseur le plus fort, et lui porte un nombre d'attaque égal au nombre de points de vie de l'unité le plus important plus deux. Il y a donc toujours au minimum 3 attaques de prévues contre une unité. Le combat peut bien sur se terminer plus tôt si l'une des deux venait à mordre la poussière ! Á la fin d'un combat, le gestionnaire de jeu teste si l'une des unités est morte (et donc la fin potentielle du jeu) et demande au joueur de la retirer de sa liste d'unités valides. Ce fonctionnement est détaillé davantage sur le diagramme de séquence 4.

3.3 Activité sur une case

Nous l'avons compris, les cases sont le nef de la guerre. Il est en effet très difficile de gagner sans contrôler les bonnes cases. Une première approche stratégique pourrait être de controler un maximum de cases par exemple ; il est donc très important d'être bien au fait de *l'état de la case*. Une case a trois états :

— Libre, c'est à dire avec aucune unité dessus.
— Contrôlé par le joueur (il dispose au moins d'une unité dessus).
— Contrôlé par l'adversaire (même condition).

L'état d'une case change suivant si un joueur la quitte ou se la fait prendre. Cet état n'est donc dépendant uniquement des déplacements et attaques des joueurs, et pas de l'état du jeu (tour actuel...). Davantage de détails sont disponibles dans le diagramme d'états-transitions 5. L'une des particularités de ce projet sont les cases hexagonales. Elles ne présentent pas de véritables soucis d'implémentation car les équations de mouvement sont facilement modélisables mathématiquement. En effet, les cases adjacentes ont pour indice (X-a, Y-b) avec a et b égaux à 0, 1 ou -1. On remarquera cependant que les cases hexagonales disposent de 6 cotés contre 4 pour les cases carrés. On peut dons s'attendre à des parties plus animées car il est plus rapide sur un plateau hexagonal de déplacer d'un coin à l'autre du plateau.

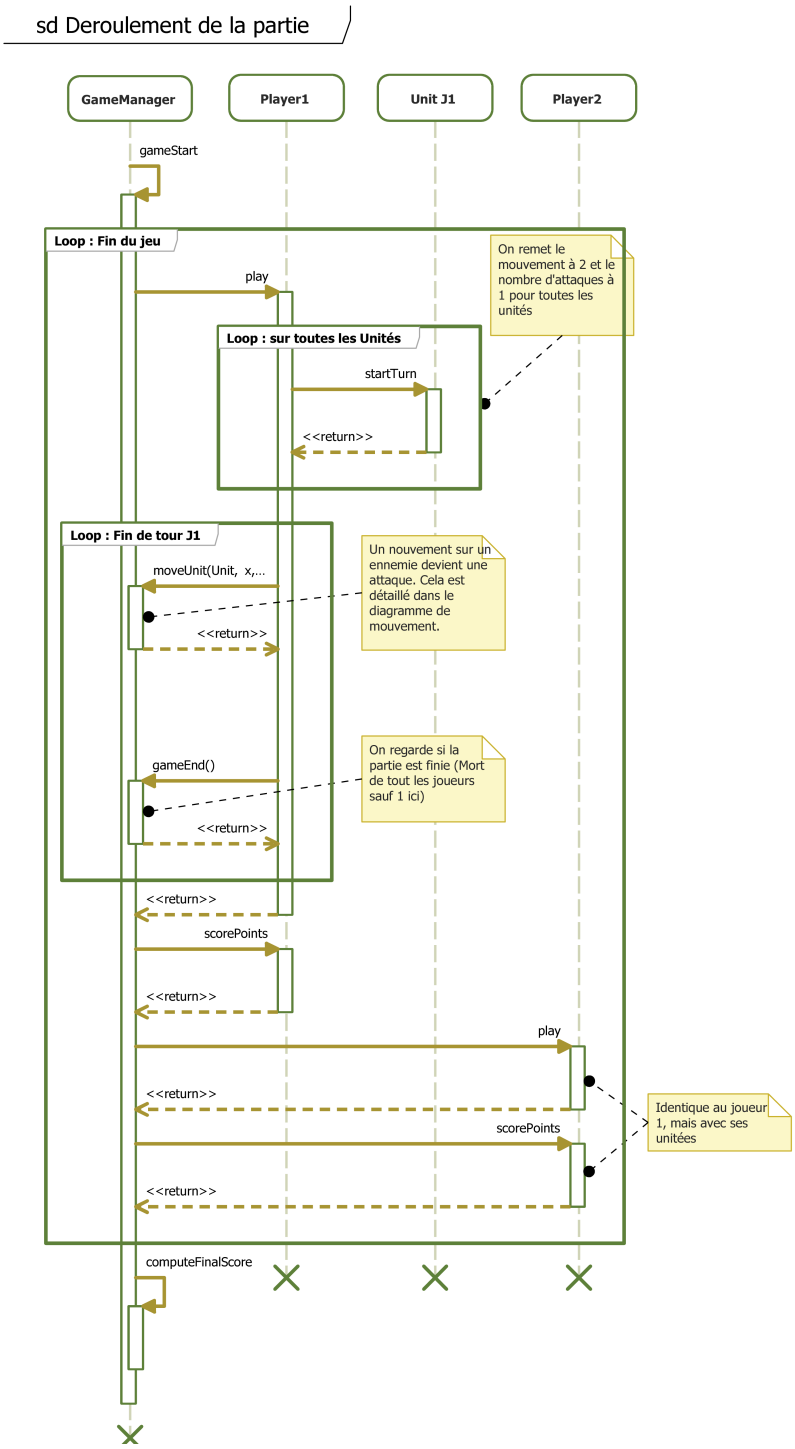

FIGURE 3 – Deroulement de la partie

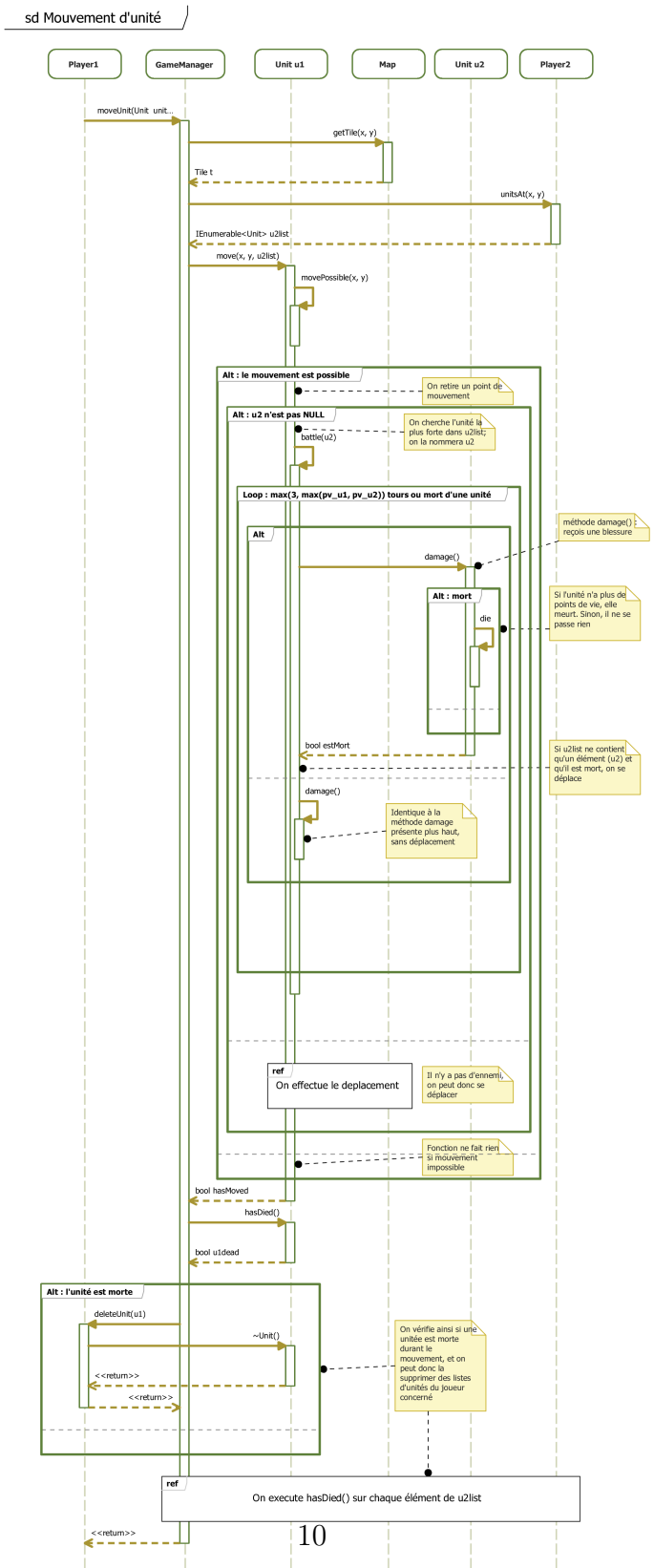

FIGURE 4 – Mouvement et combat

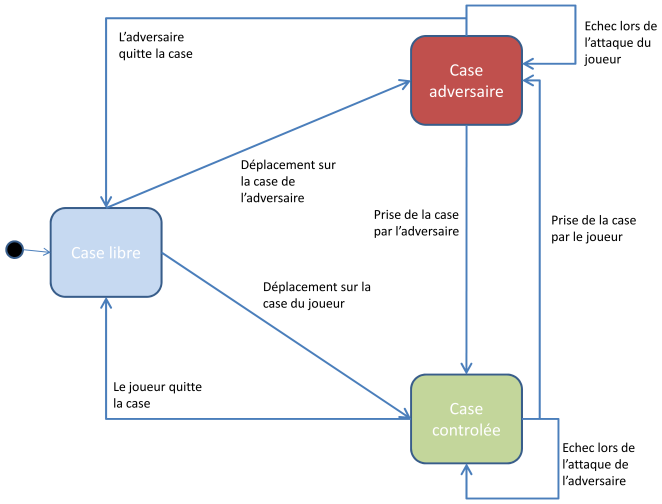

FIGURE 5 – Occupation d'une case de jeu

4 Interface utilisateur

L'interface utilisateur se présentera sous la forme de plusieurs interfaces utilisateur permettant d'accéder aux différentes fonctions du programme. On peut distinguer plusieurs grandes fonctionnalités :

— Un menu principal.
— Le plateau et l'interface de jeu.
— Un menu de sauvegarde de partie depuis le jeu.
— Un menu de chargement de partie depuis le jeu.
— Un menu de chargement de partie depuis le menu.

4.1 Menu principal

Il s'agit de l'interface que verra le joueur au lancement du jeu. Depuis ce menu, il peut créer ou charger une partie, ainsi que voir l'aide. Les paramètres de création de partie seront obtenus avec un sous-menu de creation. Les fonctionnalités de ce menu sont détaillées dans le diagramme de cas d'utilisation 6.

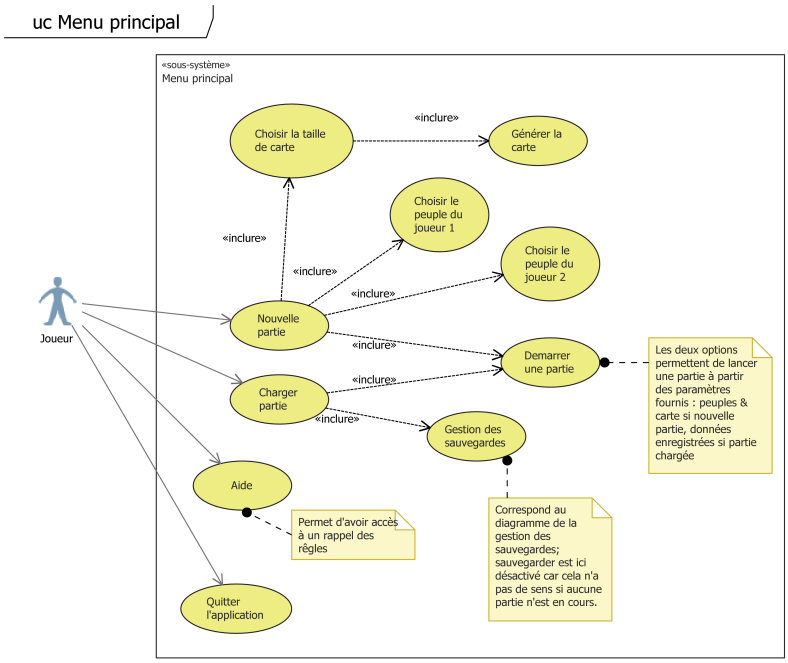

FIGURE 6 – Fonctions du menu principal

4.2 Interface durant le jeu

L'interface utilisée au cours du jeu, dite 'In-game', représente l'interface la plus vue par le joueur. C'est donc celle qui est le plus à soigner et à perfectionner. Plusieurs actions peuvent être réalisées par l'utilisateur depuis cette interface ; celles-ci sont référencées sur le diagramme de cas d'utilisation 7.

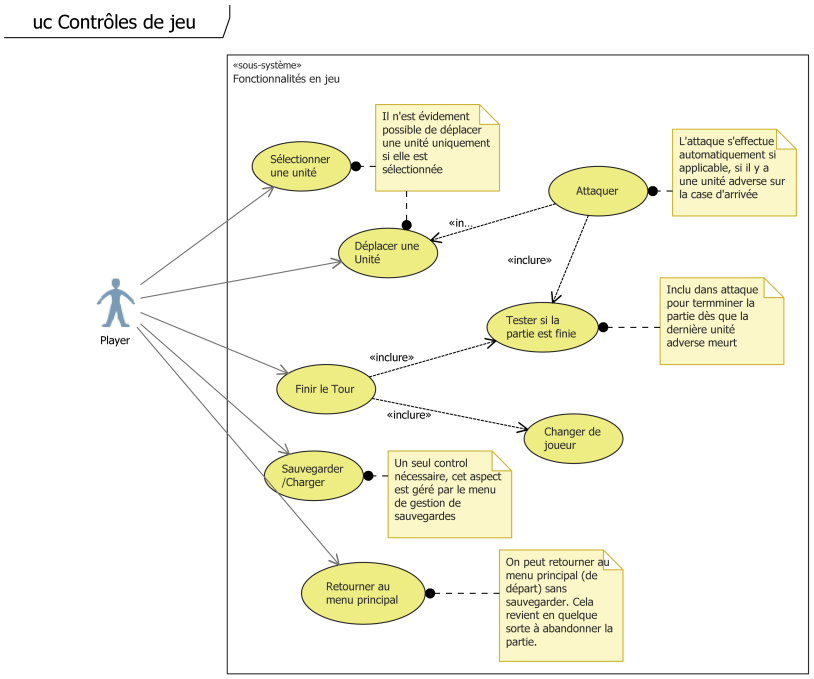

FIGURE 7 – Fonctionnalités en cours de partie

4.3 Menu de gestion des sauvegardes

Les 3 dernières fonctionnalités présentées au début de cette section sont très similaires. On se propose de les réunir en une seule interface. Ce menu regroupera donc les fonctionnalités de sauvegarde et de chargement de partie, depuis le menu et le jeu. Lorsqu'on y accède depuis le menu principal, on grisera ou cachera l'option de sauvegarde, qui n'a pas de sens ici. Ces fonctionnalités sont présentées dans le diagramme de cas d'utilisation 8.

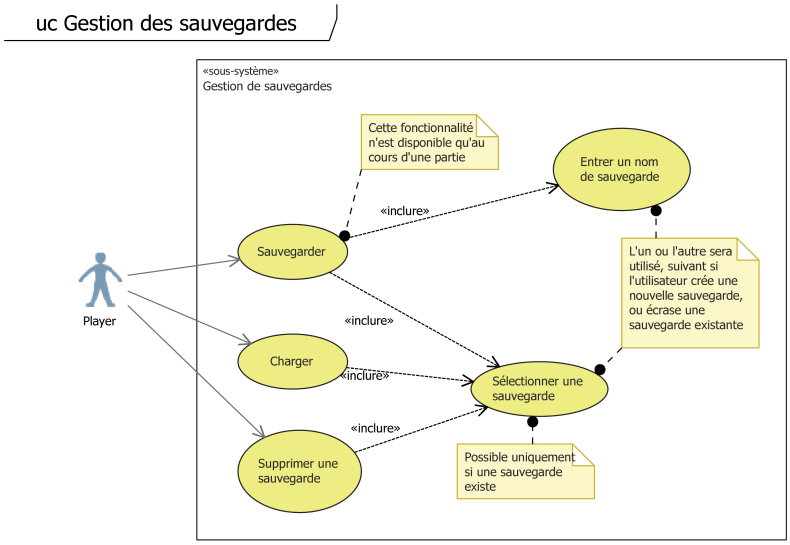

FIGURE 8 – Cas d'utilisation du menu de sauvegarde

5 Diagramme de classes

Au cours de cette étude, plusieurs problématiques ont été soulevées ; nous proposons donc un diagramme d'interfaces permettant de réponde au besoin exprimé lors de la présentation du sujet. Ce diagramme présente toutes les fonctionnalités nécessaires pour le projet. Nous y avons implémenté :

- Un monteur pour la partie ; le système permet de créer ou charger une partie et est extensible. On pourrait rajouter une monteur pour créer un GameManager avec des conditions de victoire et de défaite supplémentaires.
- Une fabrique pour les différents peuples. Il est ainsi aisé de rajouter un peuple supplémentaire.
- Un poids-mouche pour l'utilisation du plateau de jeu. L'objectif est de réduire l'empreinte mémoire de notre jeu
- Un patron de conception strategie est utilisé pour la génération avec différentes tailles de carte.
- La sauvegarde implémente elle aussi stratégie : Il serait ainsi possible d'avoir différents formats de sauvegarde.

Toutes ces informations sont détaillées sur diagramme d'interfaces 9.

Les attributs nécessaires ont alors été rajoutés pour créer les classes. Parmis les choix notables de réalisation, on citera :

- Utilisation d'une Dictionary<string, Tile> pour le poids-mouche. Le système est plus lourd que des membres statiques dans notre cas, mais il est plus extensible.
- Utilisation de la structure UnitData, pour une restauration plus facile des unités lors d'un chargement.
- Stocker les types de tiles dans une liste et non dans un tableau à deux dimensions ; il est ainsi plus facile de controler le déplacement.

Tout ces détails sont visibles sur le diagramme de classe 10.

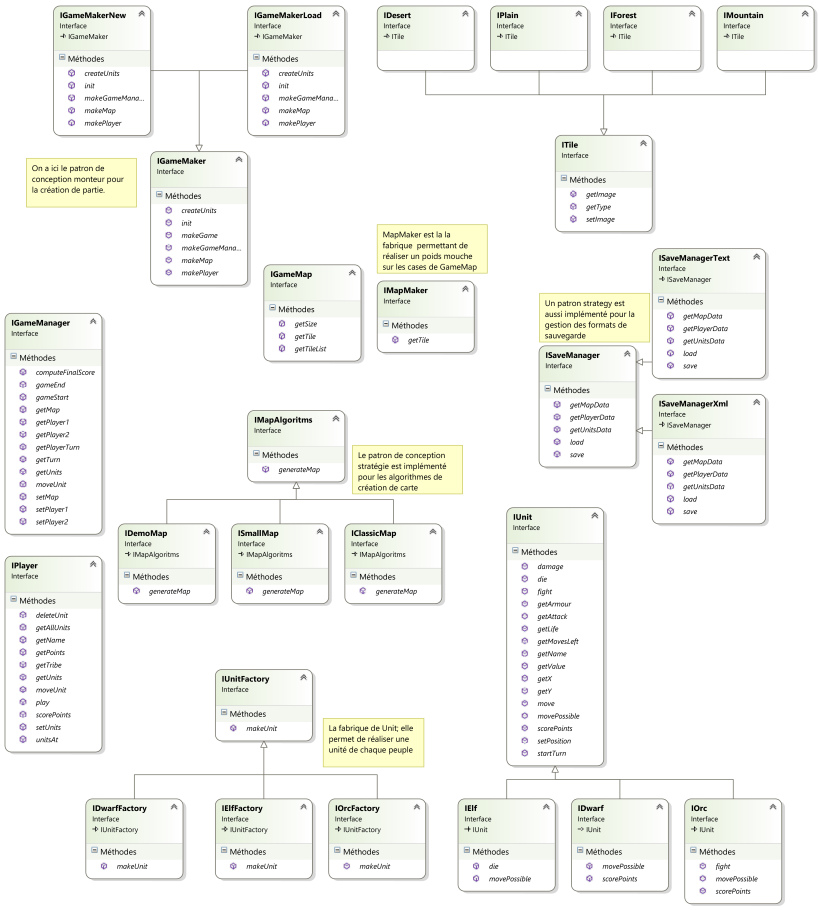

FIGURE 9 – Interfaces de programmation

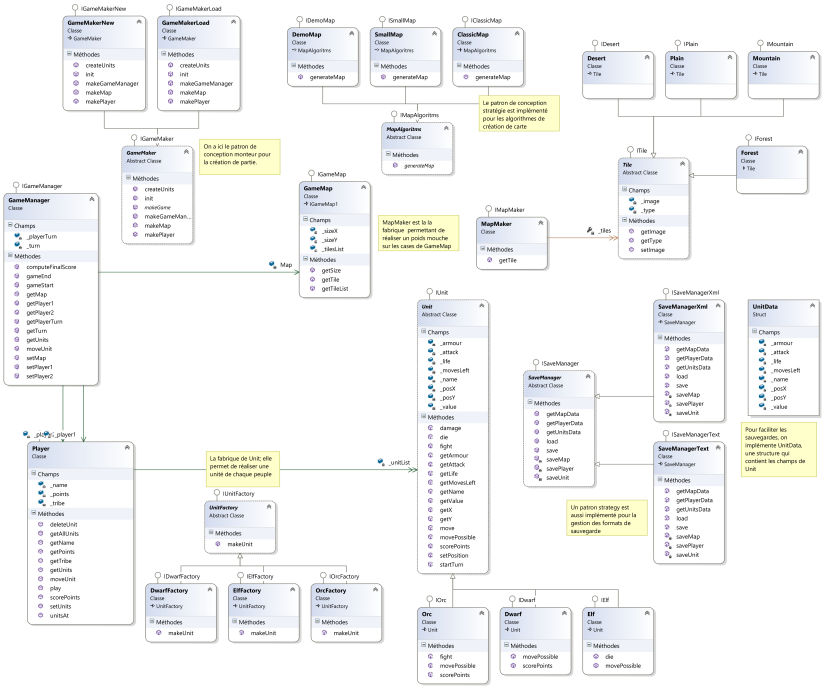

FIGURE 10 – Diagramme de classes

6 Conclusion

Cette étude nous a permis de bien nous former à l'utilisation des patrons de conception. Il ne nous reste plus qu'a démontrer leur efficacité en implémentant les classes décrites sur les diagrammes. Il faudra donc que les joueurs potentiels patientent encore un peu avant de pouvoir s'étriper joyeusement avec leurs amis !