

INSA de Rennes
Quatrième année - Informatique

Projet d'Analyse, Conception et POO

Rapport de conception

Mickaël OLIVIER, Benoit TRAVERS

18 novembre 2013

Table des matières

1	Cas d'utilisation	4
1.1	Création d'une partie	4
1.2	Tour de jeu	5
2	Diagrammes de classe	7
2.1	Modélisation globale du jeu	7
2.2	Patrons utilisés	7
3	Diagramme d'états-transitions	10
4	Diagrammes d'interaction	11
4.1	Création de partie	11
4.2	Déplacer ou attaquer une unité	12
	Appendices	14
	Annexe A Diagramme de classes	14
	Annexe B Diagramme de séquence : Création de partie	15
	Annexe C Diagramme de séquence : Déplacer ou attaquer une unité	16

Introduction

Nous avons été amenés à modéliser un jeu small world en C sharp cette année et dans ce but il nous a été donné l'occasion de modéliser le problème sous la forme de diagrammes UML. Ce rapport marque la fin de cette phase d'analyse et de conception, regroupant les différents diagrammes de cas d'utilisation, diagrammes de classe, diagramme d'états-transitions et diagrammes d'interaction.

Rappelons rapidement le principe du jeu que l'on doit implémenter : il se compose d'une seule fenêtre active que se partagent deux joueur au moyen des mêmes périphériques (en l'occurrence, clavier et souris). Cette fenêtre présente un plateau qui regroupe différents éléments : des cases, des unités, des ressources.

Au début du jeu, chaque joueur choisit un peuple parmi les trois proposés : Nains, Gaulois et Viking.

Le but du jeu est de gagner le maximum de points victoire en un certain nombre de tours de jeu défini au début de la partie. Les points de victoire se gagnent en occupant des cases ressources. Un tour consiste, pour chaque joueur, à utiliser pour chaque unité une des trois actions (déplacer, attaquer, passer).



FIGURE 1 – Small World

1 Cas d'utilisation

Premièrement, nous retrouvons un diagramme de cas d'utilisation illustrant la globalité du jeu (Figure 2). En se penchant sur ce diagramme, on remarque qu'un joueur peut créer une partie ou il peut jouer. Les diagrammes de cas d'utilisation pour la création de la partie et le tour de jeu sont décrit par la suite.

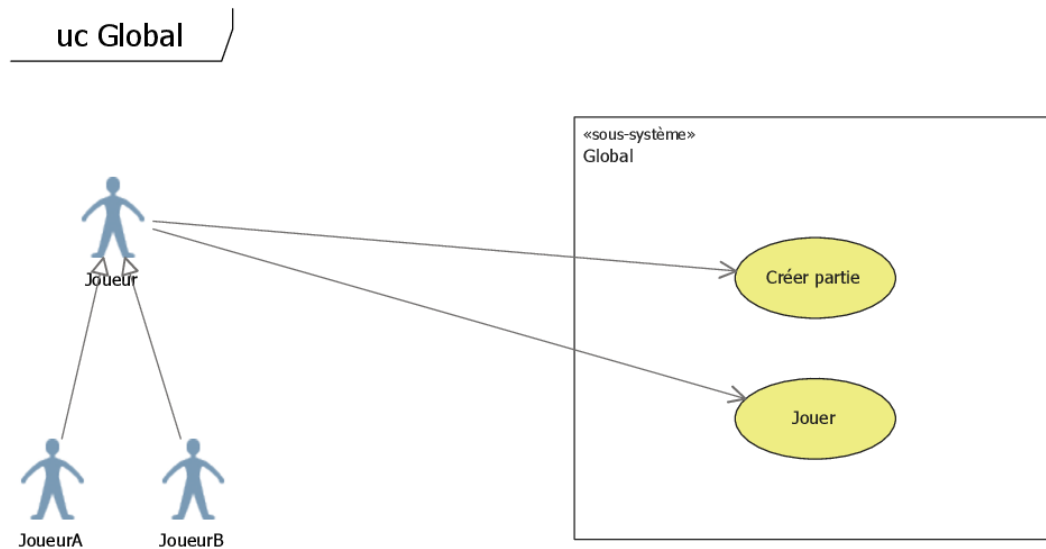


FIGURE 2 – Diagramme de cas d'utilisation global

1.1 Création d'une partie

Avant de pouvoir jouer, il faut créer la partie. Pour cela, le joueur A choisit la carte, à savoir, soit une carte de démonstration, soit une petite carte, soit une carte de taille normale. Ensuite, les joueurs devront choisir leur peuple parmi les Vikings, les Gaulois et les Nains. Ces actions sont illustrées dans le diagramme de cas d'utilisation de la Figure 3.

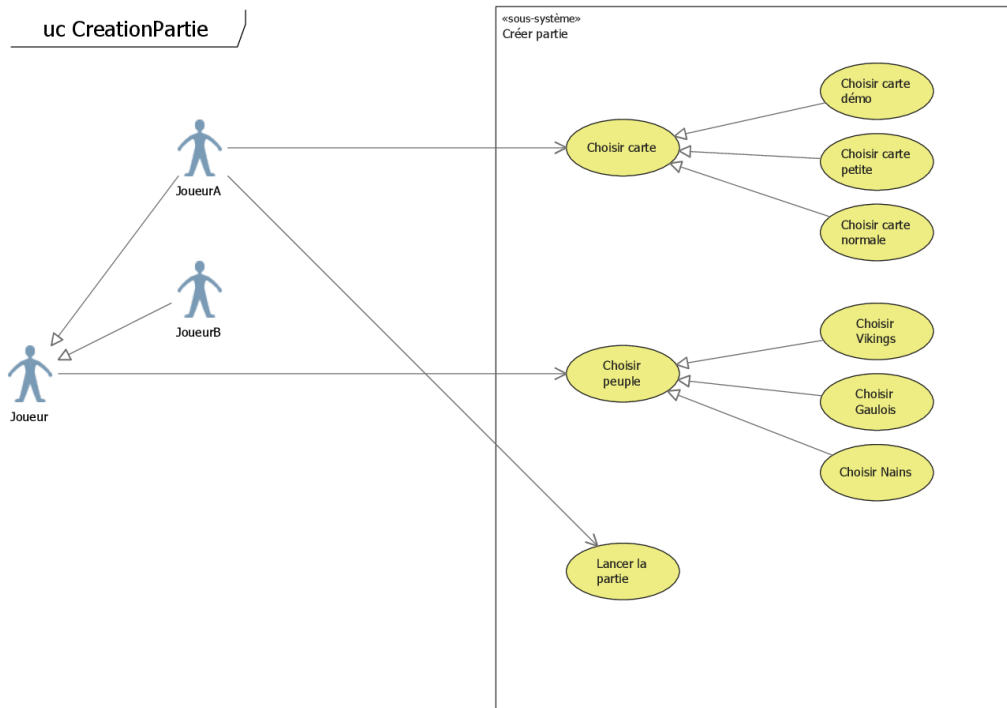


FIGURE 3 – Diagramme de cas d'utilisation : Création de la partie

1.2 Tour de jeu

Lors de son tour de jeu, un joueur peut effectuer différentes actions. Il peut déplacer une unité et pour se faire, il doit sélectionner la case contenant l'unité qu'il souhaite déplacer, il choisit l'unité sur la case sélectionnée (il peut y avoir plusieurs unités d'un même joueur sur une même case) puis il sélectionne une case de destination.

Un joueur peut aussi attaquer une unité. Pour cela, le joueur doit sélectionner la case contenant l'unité qu'il souhaite engager au combat, il choisit l'unité sur la case sélectionnée puis il sélectionne la case de destination avec l'unité adverse qu'il souhaite attaquer.

Enfin, un joueur peut passer le tour d'une unité en sélectionnant la case contenant l'unité puis en choisissant l'unité sur cette case ou il peut simplement passer son tour. Ces actions sont illustrées dans le diagramme de cas d'utilisation de la Figure 4.

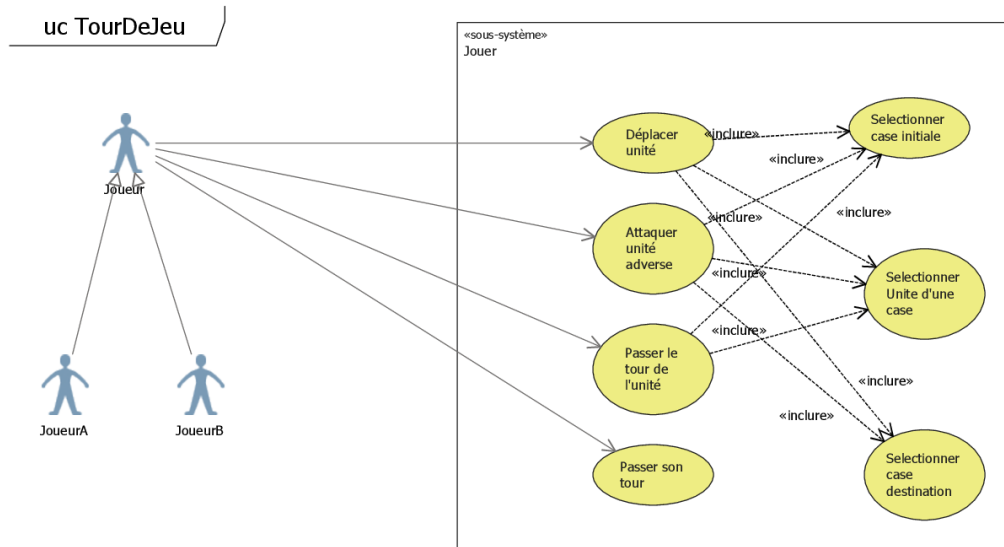


FIGURE 4 – Diagramme de cas d'utilisation : Tour de Jeu

2 Diagrammes de classe

2.1 Modélisation globale du jeu

En annexe A, on retrouve le diagramme de classe de notre jeu avec les différents patrons utilisés. Par la suite, nous allons détailler les différents patrons utilisés.

2.2 Patrons utilisés

Le premier patron utilisé est la fabrique. La classe `Peuple` est considérée comme une fabrique d'Unité.

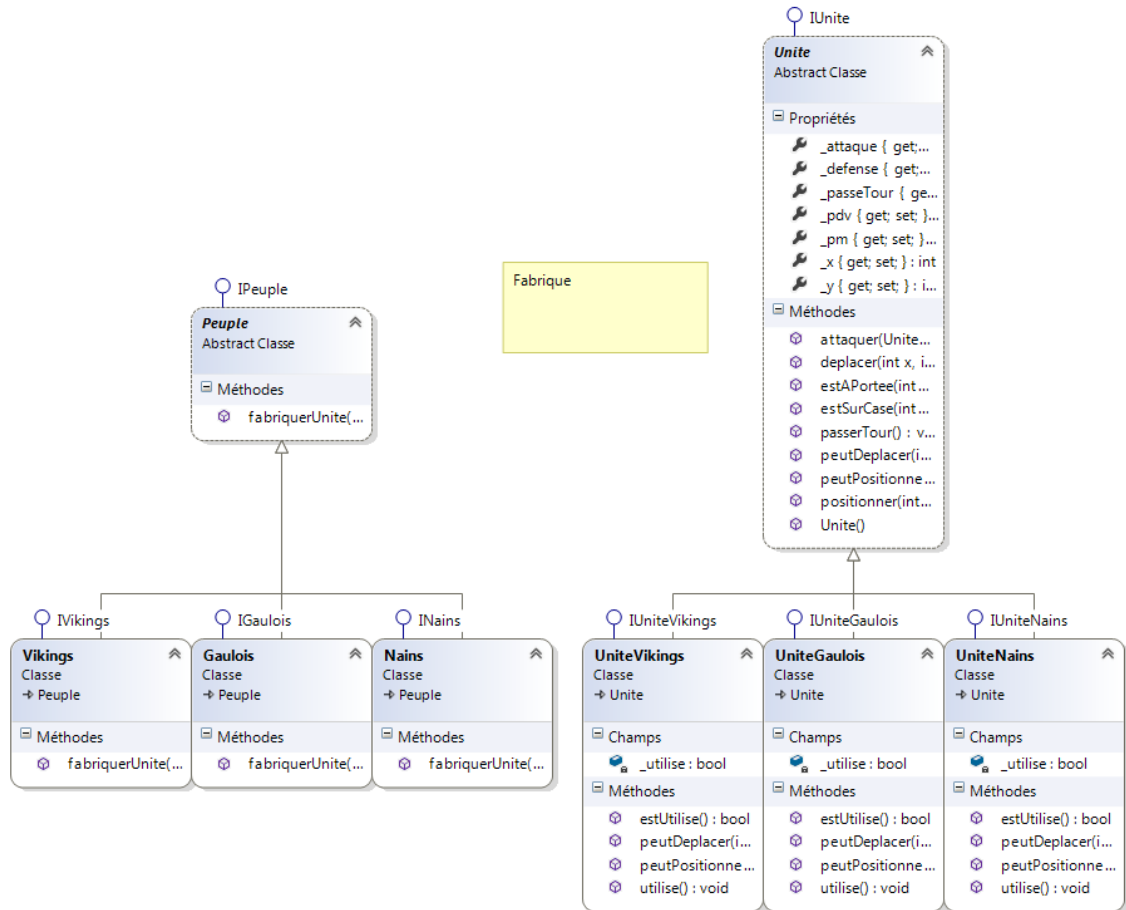


FIGURE 5 – Fabrique

La deuxième patron utilisé est le monteur. Ce patron permet l'assemblage des différents composants de la partie.

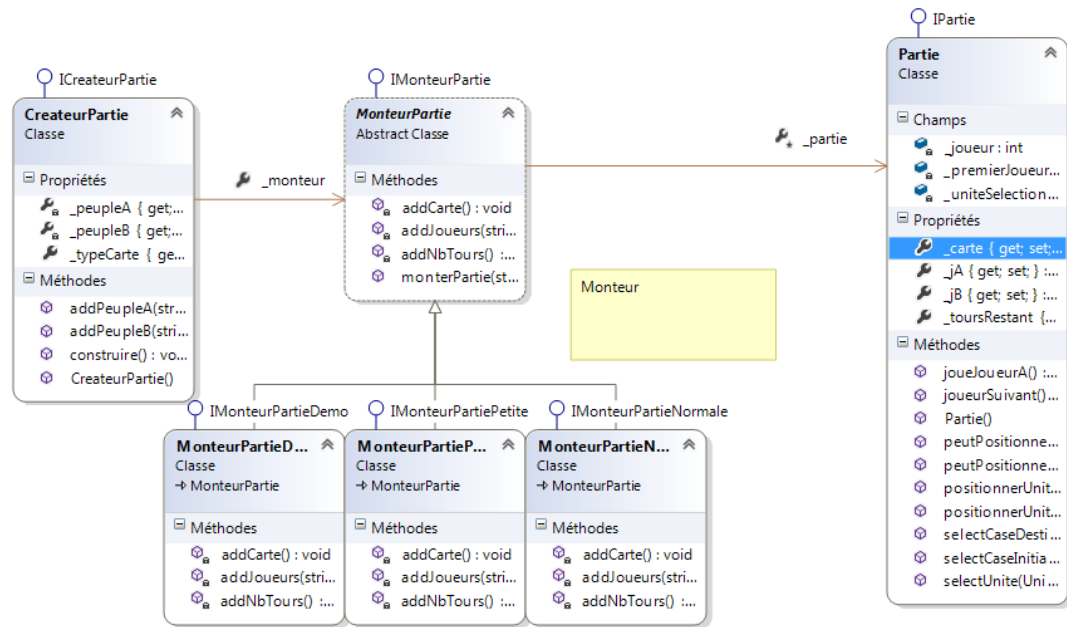


FIGURE 6 – Monteur

Un autre patron utilisé est le poids-mouche. La classe **Carte** contient un tableau de **Case**. Cependant chaque case n'est instanciée qu'une seule fois par la classe **FabriqueCase** dans l'objectif de ne pas consommer trop de mémoire.

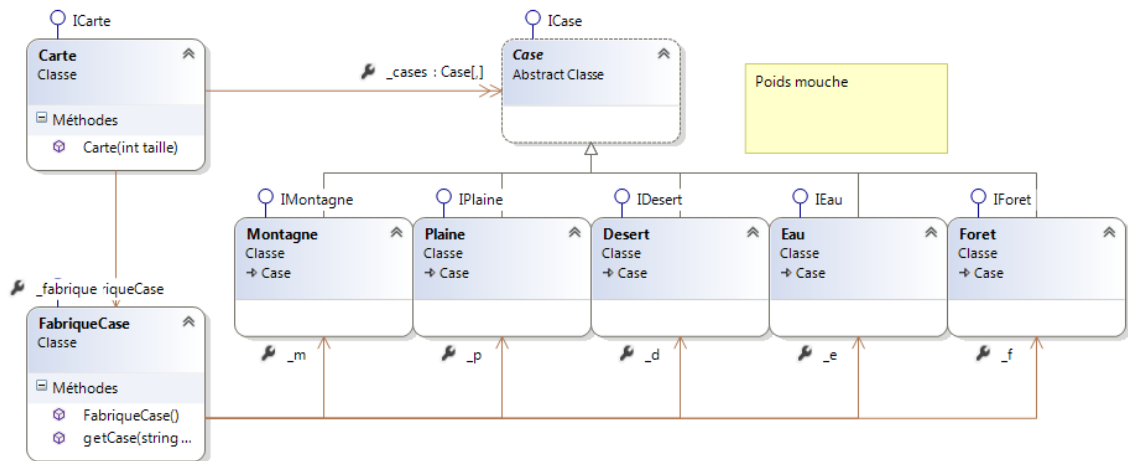


FIGURE 7 – Poids-Mouche

Le dernier patron utilisé est la stratégie. Le créateur carte va faire appel à l'algorithme de création de carte développé en C++. Mais l'appel à cet algorithme sera différent selon le type de carte à créer. (On rappelle que l'on considère dans notre implémentation trois types de cartes : Démo, Petite et Normale).

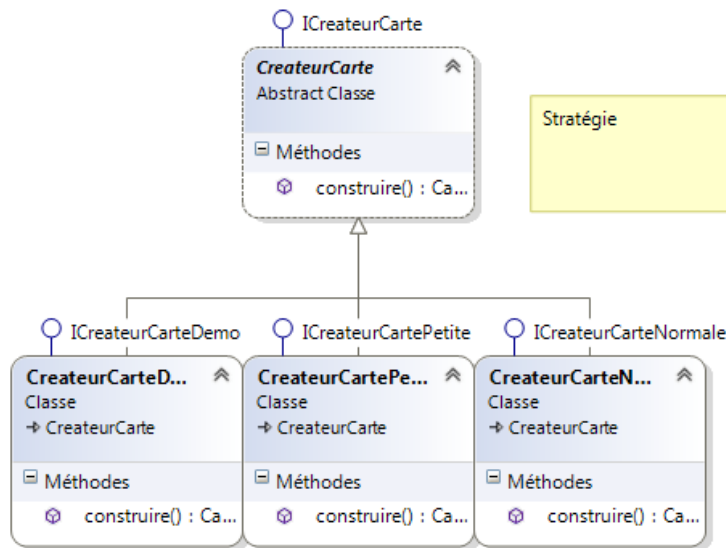


FIGURE 8 – Stratégie

3 Diagramme d'états-transitions

Le diagramme d'états-transitions représenté Figure 9 illustre le cycle de vie d'une unité. Pour commencer, une unité est positionnée sur la carte. Puis durant son tour de jeu, elle pourra soit être sélectionnée puis ensuite être déplacée, attaquer ou passer son tour, soit ne rien faire. Lorsque l'adversaire joue, une unité pourra se faire attaquer, dans ce cas, elle rentre dans l'état Défense.

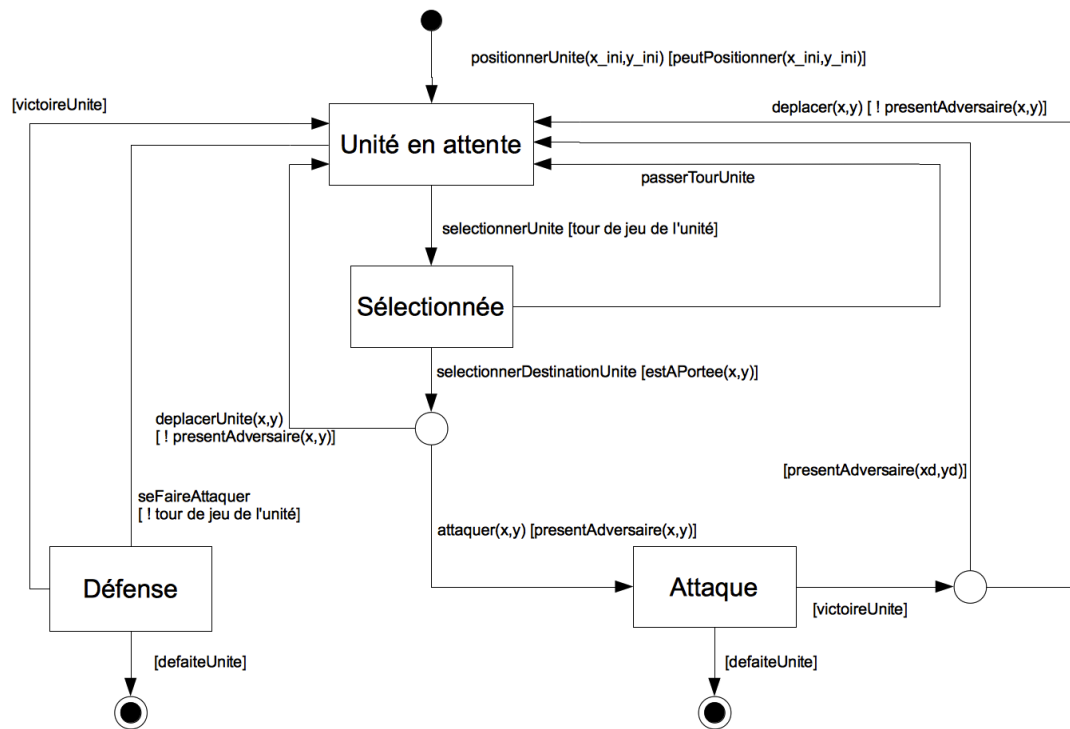


FIGURE 9 – Diagramme d'états-transitions

4 Diagrammes d'interaction

4.1 Création de partie

En annexe B, on retrouve le diagramme de séquence modélisant la création d'une partie. Pour commencer, le joueur A sélectionne la carte et son peuple, puis le joueur B sélectionne son peuple avant que le joueur A lance la création de la partie. La Figure 10 montre les actions effectuées lors de la sélection du peuple par le joueur A pour que le joueur B ne puisse sélectionner le même peuple.

Puis lors du montage de la partie, trois cas vont se présenter : il faut créer une partie de démonstration, une petite partie ou une partie normale. Pour chaque cas, on va utiliser le monte adéquat puis lancer le processus de création de partie. La Figure 11 montre les actions effectuées pour créer et ajouter la carte.

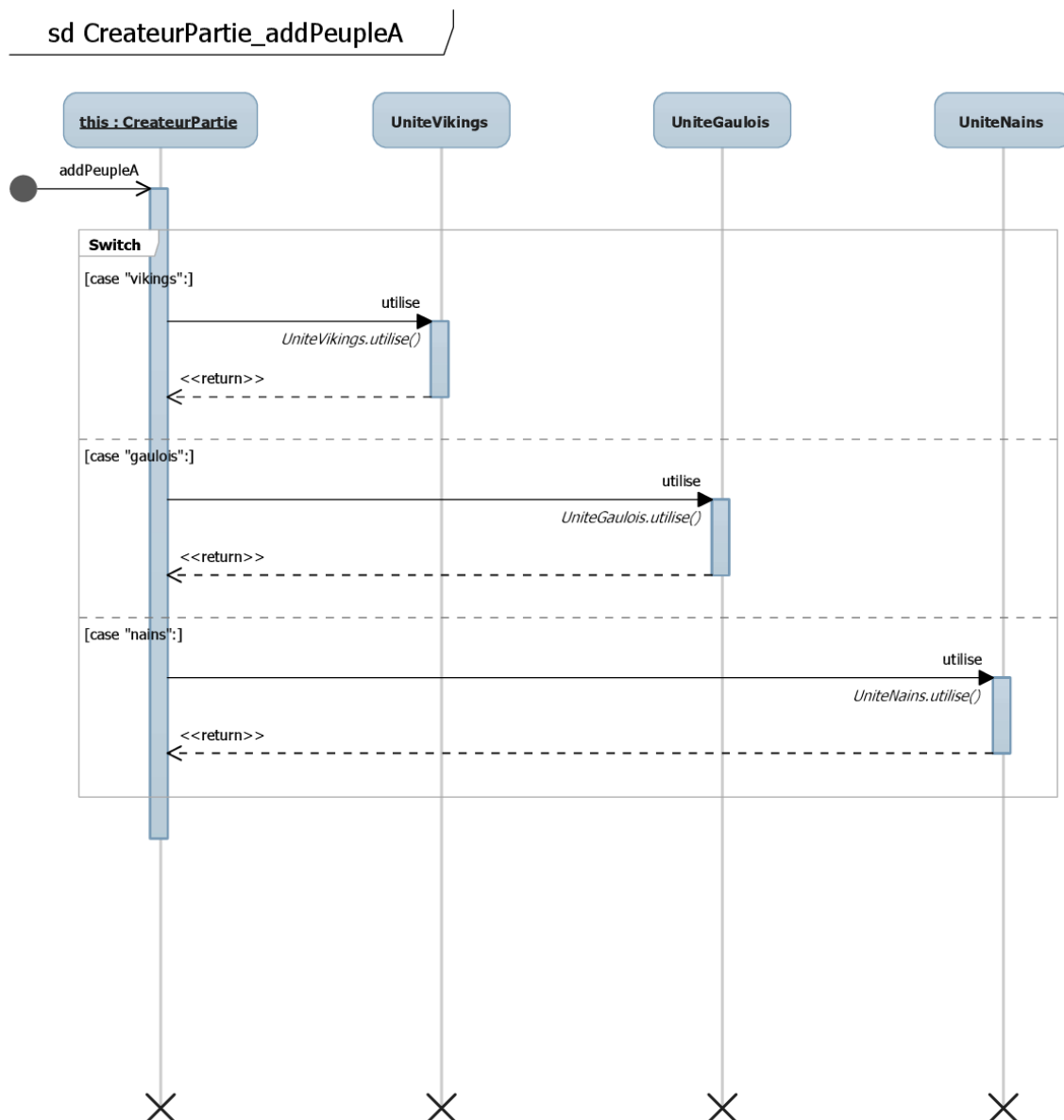


FIGURE 10 – Diagramme de séquence : Ajouter le peuple du joueur A

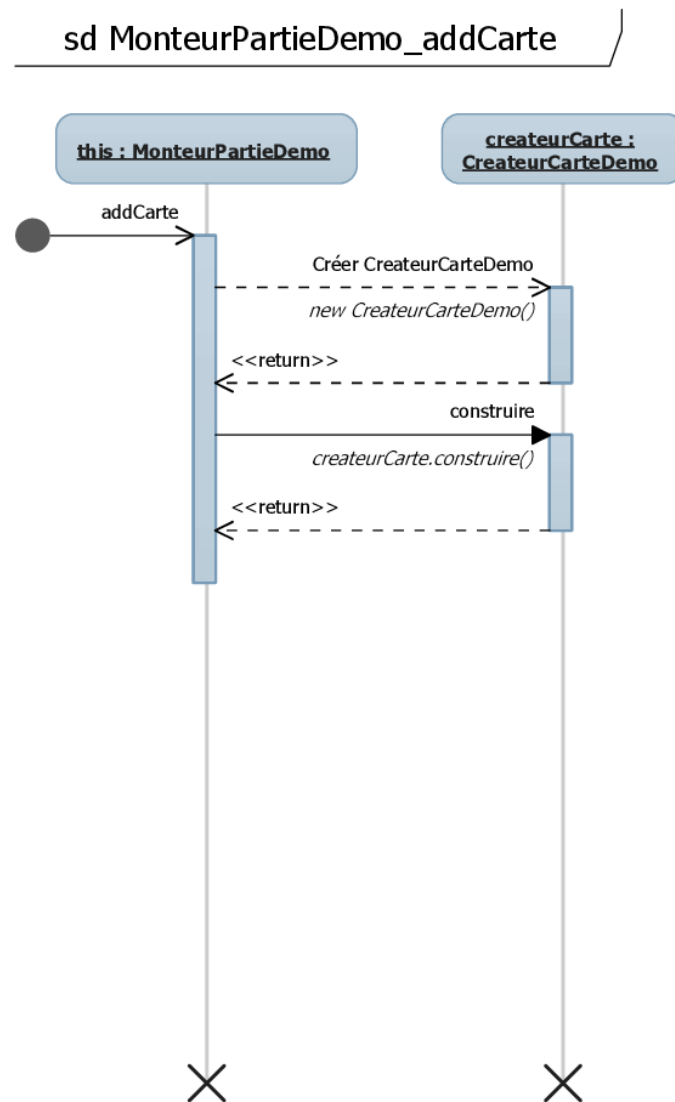


FIGURE 11 – Diagramme de séquence : Ajouter la carte

4.2 Déplacer ou attaquer une unité

En annexe C, on retrouve le diagramme de séquence modélisant le déplacement ou le lancement d'un combat d'une unité. Pour commencer, le joueur sélectionne une case. Si ce joueur n'a pas d'unité sur cette case, il ne pourra pas poursuivre cette action.

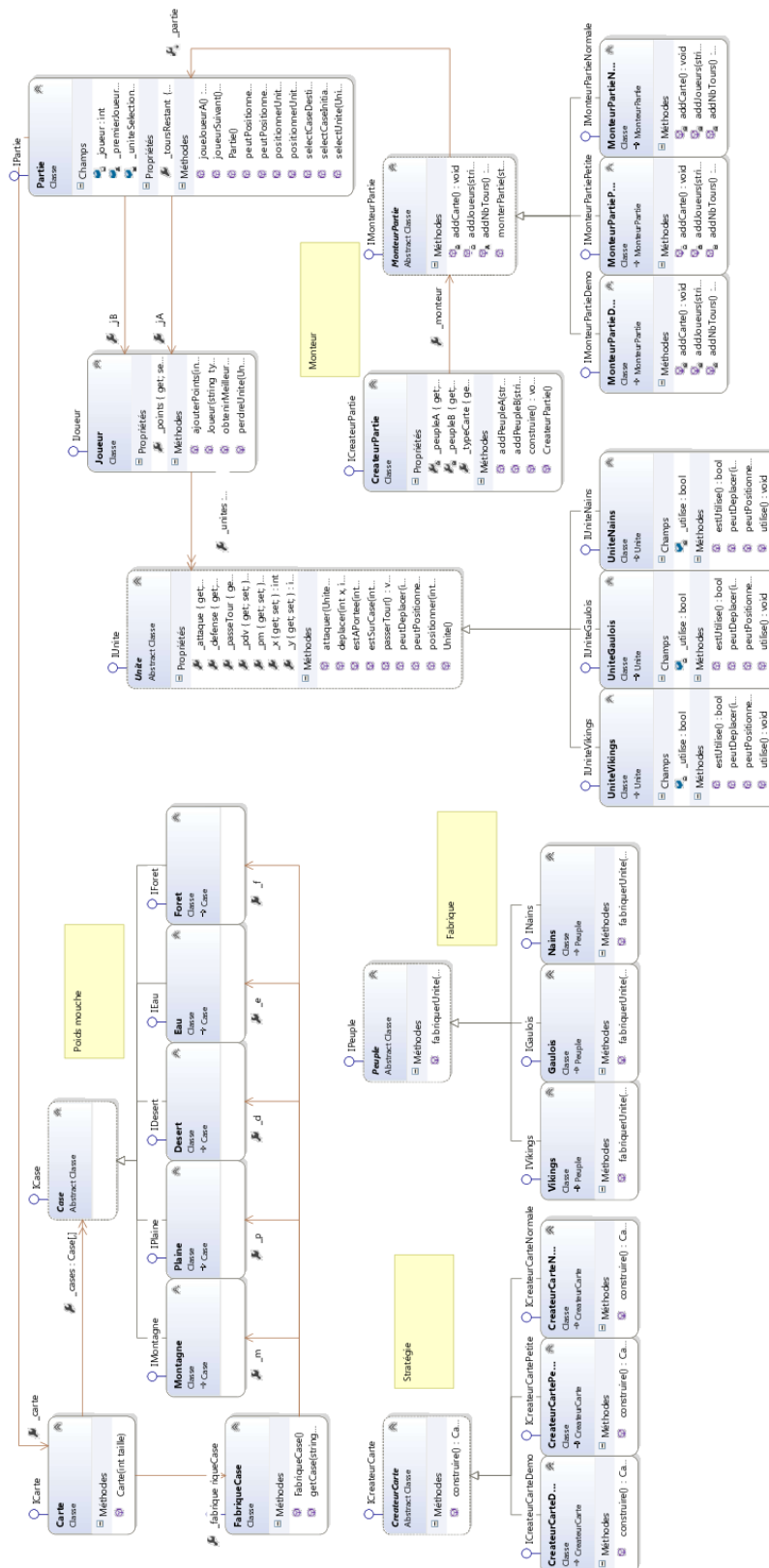
Si, au contraire, le joueur a au moins une unité sur la case sélectionnée, il devra choisir l'unité à sélectionner et qui sera conservé par l'attribut `_unitéSélectionnée` de la classe `Partie`. Ensuite, le joueur devra sélectionner une case de destination. On vérifie si cette case est à portée de l'unité sélectionnée, c'est-à-dire si il existe un chemin pour aller jusqu'à cette case sans dépasser le nombre de point de mouvement de l'unité.

Ensuite, soit la case de destination sélectionnée n'a pas d'unités ennemies et l'unité sélectionnée va se déplacer sur cette case, soit il y a au moins une unité ennemie et l'unité sélectionnée attaque la meilleure unité ennemie présente sur la case.

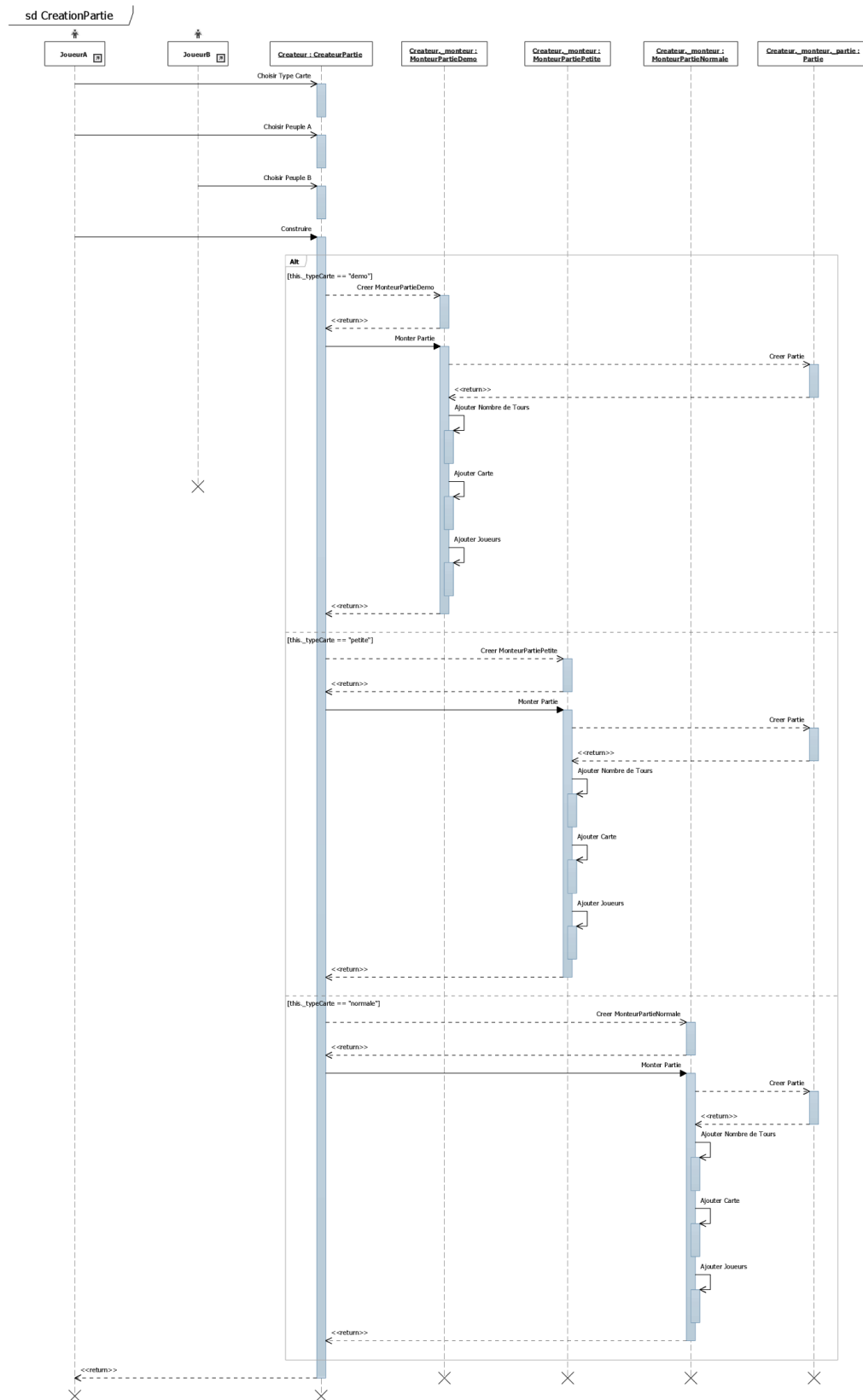
Conclusion

Ce rapport nous donne une idée de la modélisation de notre jeu. Cependant notre modèle n'est pas définitif, il évoluera par la suite avec la phase d'implémentation.

Annexe A Diagramme de classes



Annexe B Diagramme de séquence : Création de partie



Annexe C Diagramme de séquence : Déplacer ou attaquer une unité

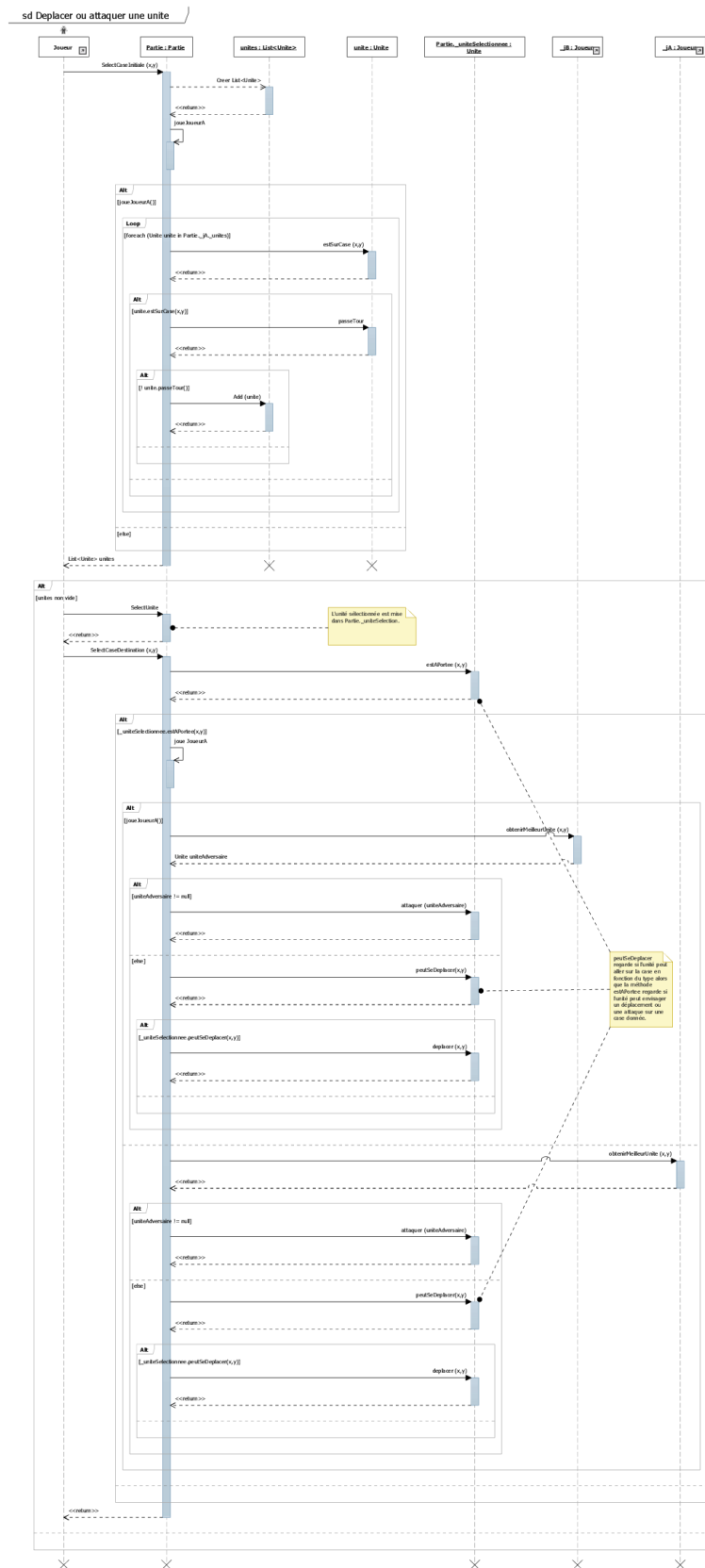


Table des figures

1	Small World	3
2	Diagramme de cas d'utilisation global	4
3	Diagramme de cas d'utilisation : Création de la partie	5
4	Diagramme de cas d'utilisation : Tour de Jeu	6
5	Fabrique	7
6	Monteur	8
7	Poids-Mouche	8
8	Stratégie	9
9	Diagramme d'états-transitions	10
10	Diagramme de séquence : Ajouter le peuple du joueur A	11
11	Diagramme de séquence : Ajouter la carte	12