Projet Modélisation et Programmation Orientées Objet
Rapport de conception
Quatrième année - informatique

Alexandre Audinot, Dan Seeruttun--Marie

12/11/2014

Table des matières

1 Introduction **3**

2 Généralités sur le jeu **4**

3 Approche en détail **5**

3.1 Description des cas d'utilisation 5

3.1.1 Vue d'ensemble 5

3.1.2 Déplacement 6

3.1.3 Combat 6

4 Structure du code **16**

5 Conclusion **17**

1 Introduction

2 Généralités sur le jeu

3 Approche en détail

Dans cette partie, nous allons visualiser toutes les cas possibles du jeu, utilisant des diagrammes d'état transition.

3.1 Description des cas d'utilisation

Dans cette partie, nous allons visualiser toutes les cas possibles du jeu, utilisant des diagrammes d'état transition.

3.1.1 Vue d'ensemble

Ce diagramme permet de situer les différents cas possibles de manière générale. Le détail sera fait dans les parties suivantes.

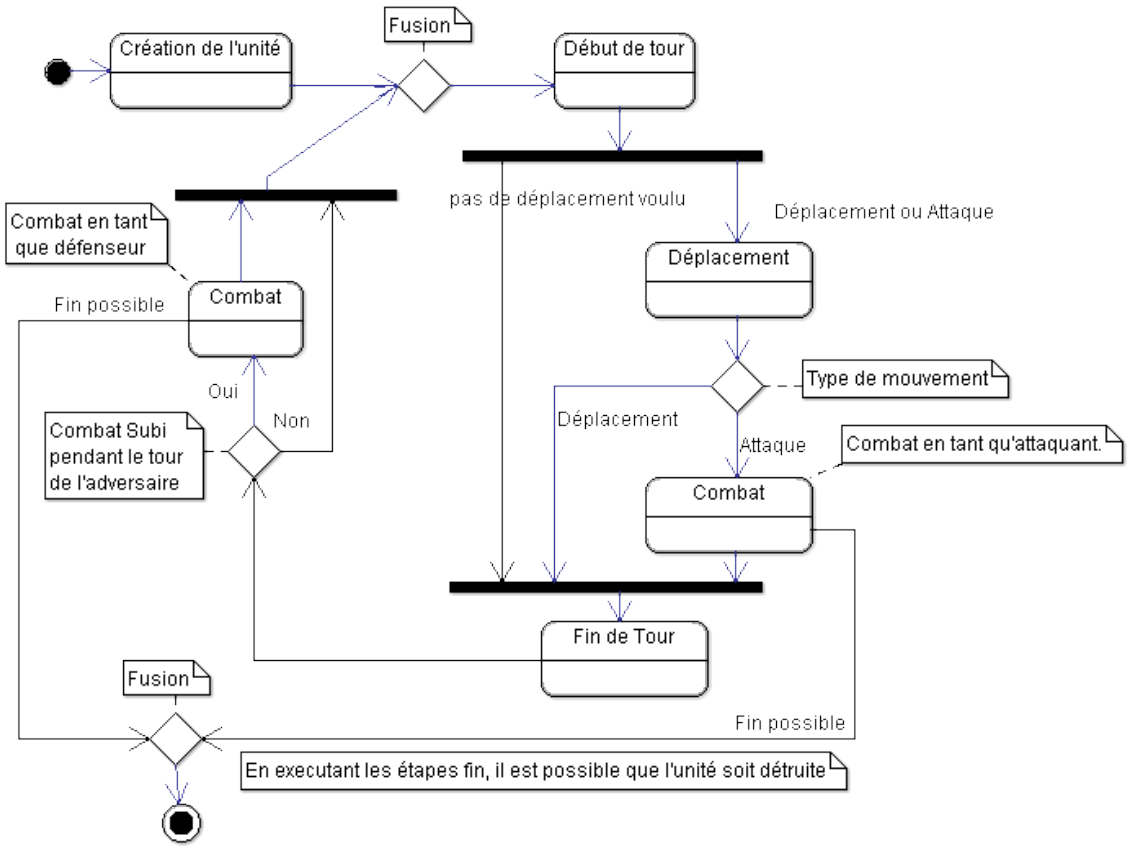

FIGURE 1 – Vue d'ensemble

Le cycle de l'unité est le suivant : elle est crée, puis commence son tour, elle peut se déplacer sur une case vide (Déplacement) ou attaquer (Déplacement + Combat). Dans le

cas d'un combat, on devra rechercher un défenseur sur la case attaquée. Si aucun n'est trouvé, l'API enverra une erreur. Dans le cas de plusieurs unité présentes sur la case, on choisira l'unité de plus grande défense, et en cas d'égalité, on la choisira au hasard. Cependant, l'unité peut avoir des bonus quelle récupérera en fin de combat (Fin de combat). Après, elle finit son tour, pour en commencer un autre ensuite. Il est possible que l'unité soit attaquée hors de son tour, ainsi, Combat et Fin de combat sont de nouveau appelés pour la défense de l'unité. À chaque combat mené, une unité peut être appelée à perdre des points de vie. Dans le cas de perte de tous les points de vie, l'unité est détruite.

3.1.2 Déplacement

Ce diagramme permet de représenter toutes les possibilités de déplacement des différentes unités. On définit la variable d qui représente le coût du déplacement sur la case courante. On définit la constante D qui représente le déplacement autorisé pour chaque pièce en début de tour. Dans l'implémentation proposée, D vaut 2 au premier déplacement de l'unité.

On calcule pour commencer la valeur du coût de déplacement sur la case courante d. Pour cela, on teste le type de l'unité, puis le type de case où elle est pour en déduire d. Par exemple, si l'unité est un orc, qui se déplace sur une plaine, le coût de déplacement d n'est que de 0.5. Il est important de noter qu'un nain qui veut se déplacer sur une case montagne aura un coût de déplacement nul.

Après le calcul de d, on vérifie si le déplacement est possible en soustrayant d à D. Le résultat ne doit pas être inférieur à 0. Dans ce cas, le déplacement n'est pas possible et l'API affichera un message d'erreur. Sinon, le déplacement aura lieu, puis on définira D -= d, pour un possible nouveau déplacement.

3.1.3 Combat

Ce diagramme permet de représenter toutes les possibilités de combat entre 2 unités.

Premièrement, on calcule le nombre de tours de combats nécessaires maximum nbCombat = Max(3,X+2), X étant le nombre de points de vie maximum des unités au combat. Ensuite, on calcule les probabilités de combat. On tire ensuite un nombre au hasard entre 1 et 100. Si le nombre appartient à la tranche du joueur ciblé (35

subsubsectionCalcul de probabilités de combat La première chose réalisée est le calcul de 2 rapports de force entre les pièces. Ils concernent respectivement l'attaque de la première pièce et la défense de la deuxième pièce, et inversement. Par exemple, si l'attaque et la défense analysée sont égales, alors le rapport de force vaudra 50Par exemple, Si l'attaquant a 4 en attaque et l'attaqué a 4 en défense (en tenant compte des bonus de terrain et du nombre de points de vie restant), l'attaquant à 50de malchance de prendre des dégats. S'il a 3 att. contre 4 déf., le rapport de force est de 75= 25

subsubsectionFin de combat

Ce diagramme représente les possibilités de jeu en fin de combat.

En cas de destruction d'unité à la suite d'un combat, qu'il soit défensif ou attaquant, l'unité peut avoir droit à des bonus/malus, selon son type et la case sur laquelle il a gagné le combat. Par exemple, une unité nain ne peut faire gagner de points de victoire. Dans tous les autres cas, toutes les unités gagnent un point de victoire. Il leur est ensuite possible de

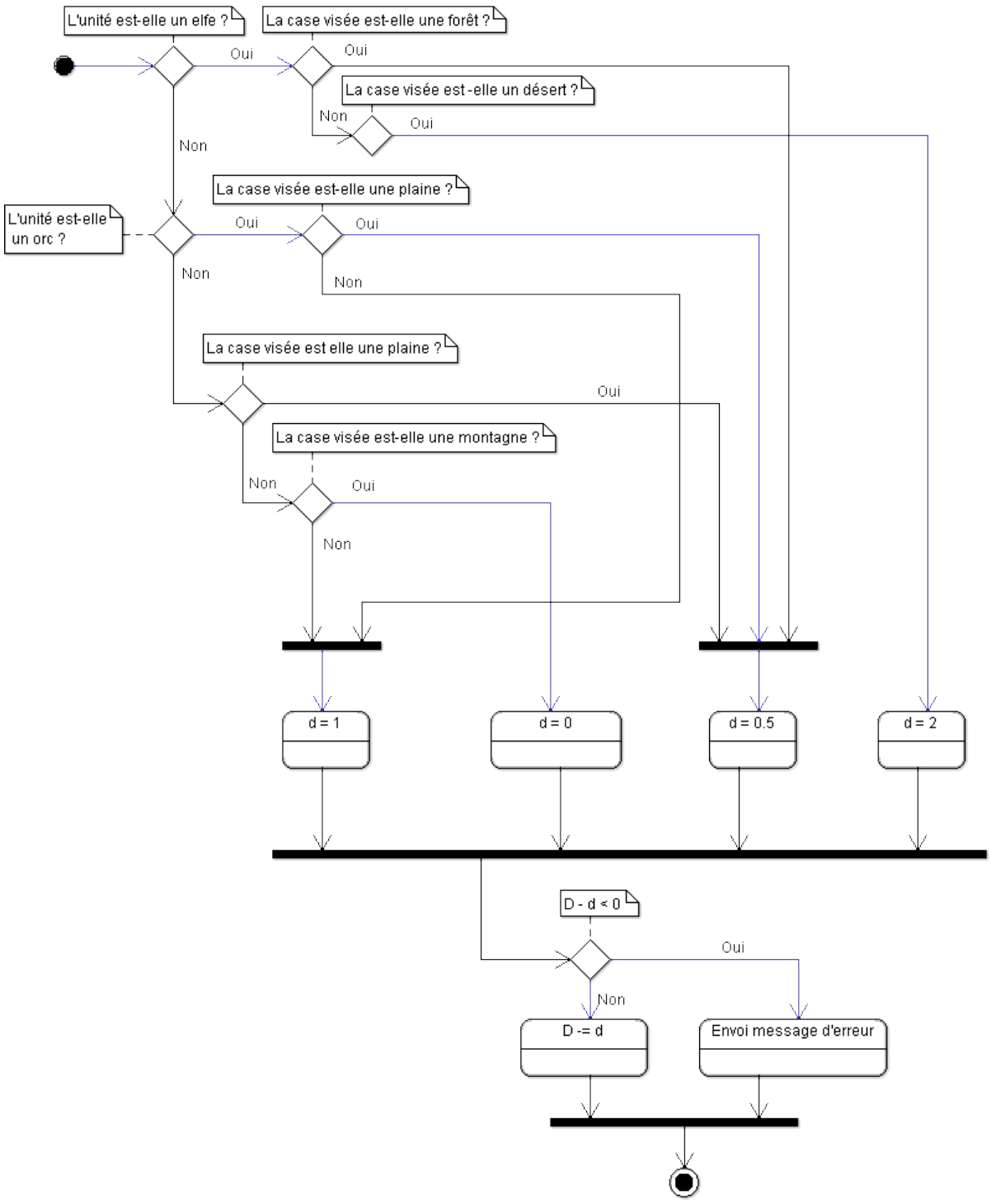

FIGURE 2 – Le déplacement d'une pièce

gagner un autre point de victoire, si l'unité est un est un orc placé sur une case Forest, un

point de victoire lui est attaché. Ces points sont cumulables et seront ajoutés au total de points de victoire en fin de partie. Attention, si l'unité orc meurt, le point de victoire bonus disparaît avec lui.

subsectionInteractions entre les différentes composantes du jeu Dans cette partie, nous allons présenter par des diagrammes d"interaction comment fonctionne plus profondément le jeu. Nous commencerons par la procédure d'initialisation du jeu.

subsubsectionInitialisation

Ce diagramme de séquence permet de différencier la charge d'une partie de la création d'une partie. On voit qu'on ne peut pas charger et créer en même temps. Le Groupe de classes pour init est décrit dans les prochains diagrammes.

Dans ce diagramme de séquence, on peut voir l'ordre de création des objets. CreateGame initialise Player, puis le Board, puis World, puis IUnit, qui prend en charge Unit.

Img InitBoard Dans ce diagramme de séquence, on peut voir l'ordre de création du Board, avec l'appel à une interface, qui apprelera un monteur qui créera les cases pour l'Abstract-Board, qui initialisera les cases Tiles et leur Position associée.

subsubsectionCombats et déplacements Ce diagramme de séquence présente les actions possibles.

Une unité peut soit gagner le combat, ou le perdre ou faire match nul. Dans le cas de la défaite, on tue l'unité si ses hp sont plus petits que 0.Sinon, comme dans le cas du match nul, on ne fait rien. Dans le cas de la victoire, on bouge sur la case de combat si elle est libre, et on fait les traitements de victoire. Une unité peut aussi bouger. Un joueur perd la partie s'il n'a plus aucune unité.

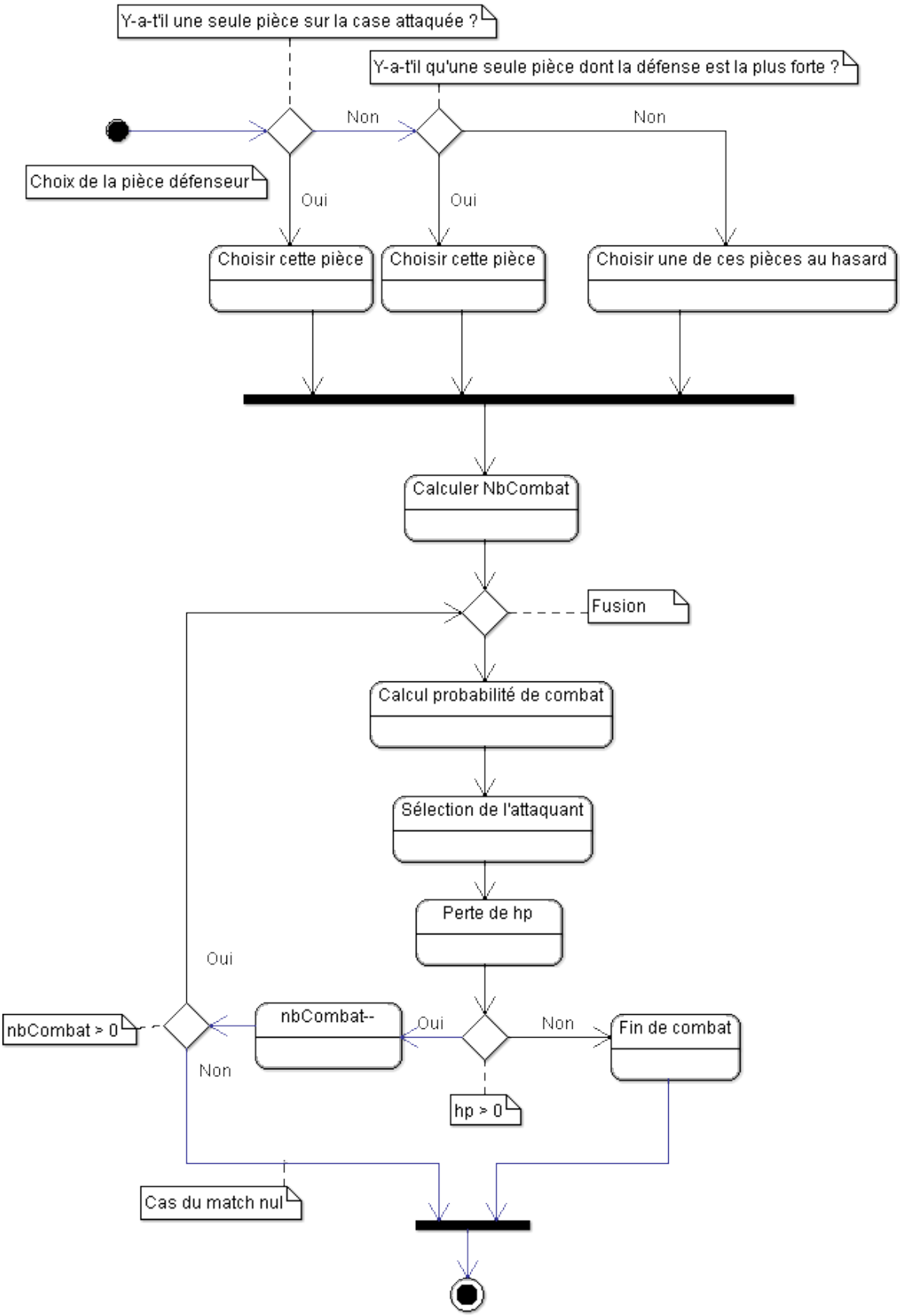

FIGURE 3 – Les combats d'une pièce

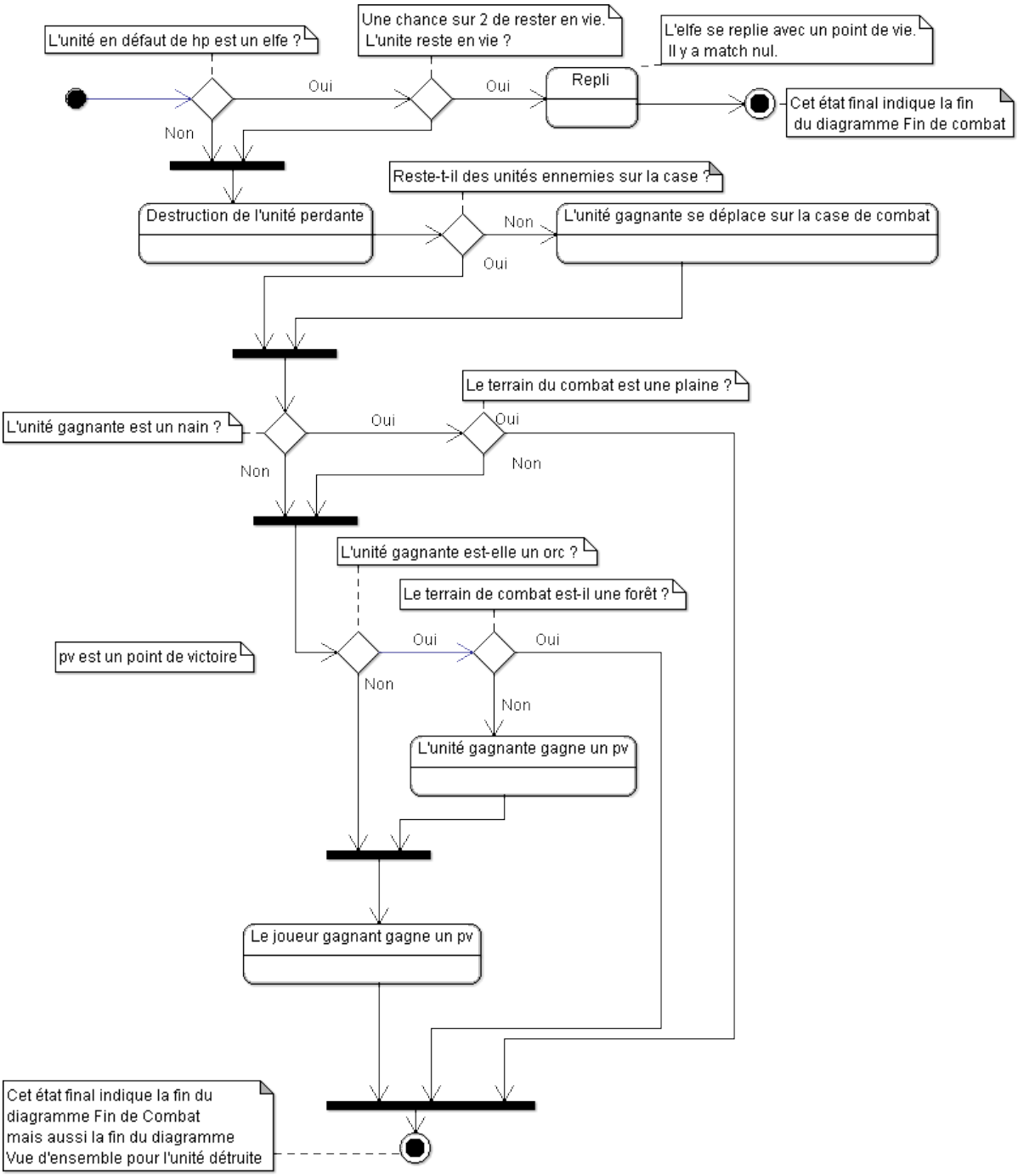

FIGURE 4 – Les cas de fin de combat d'une pièce

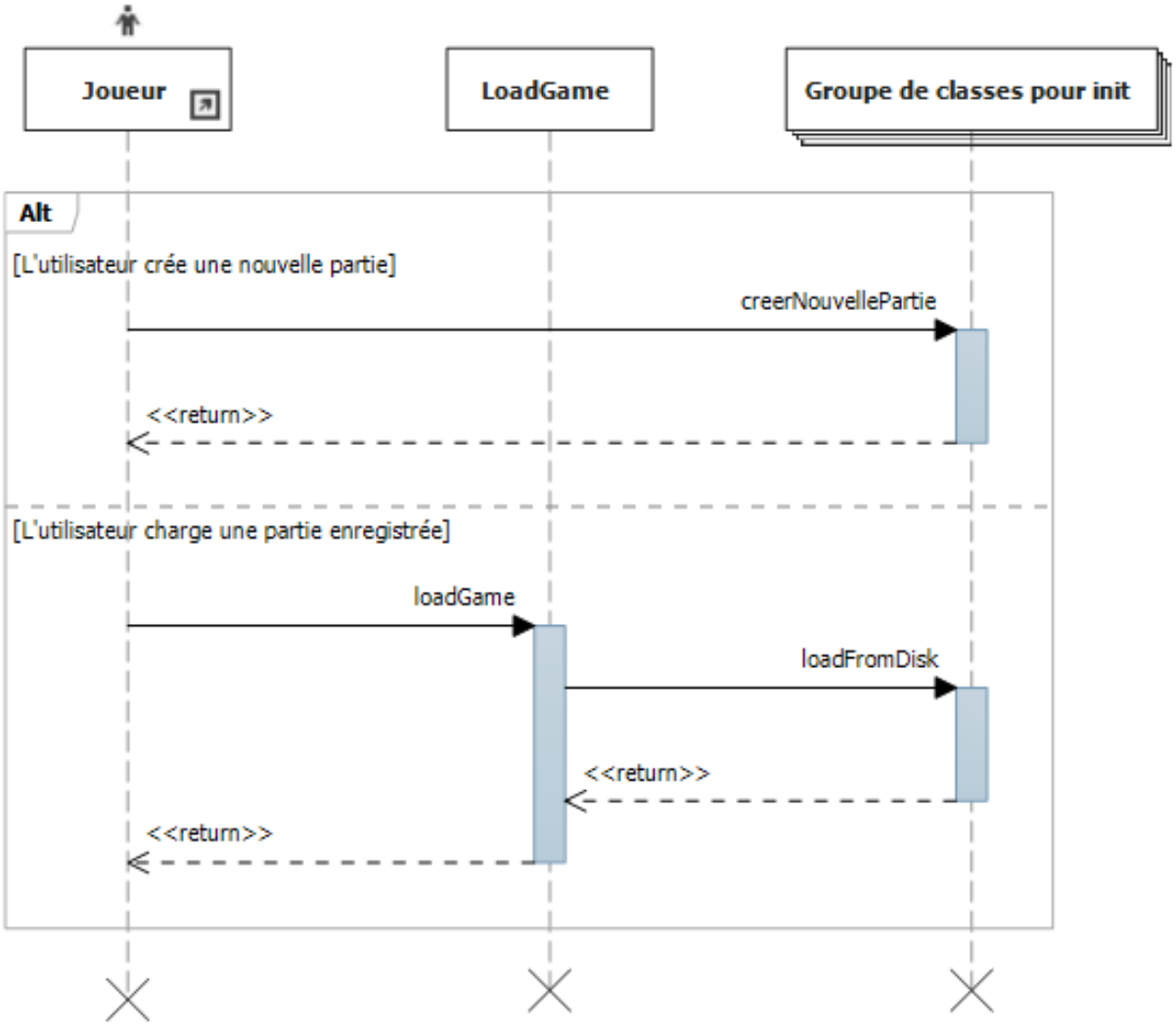

FIGURE 5 – Chargement du jeu par diagramme de séquence

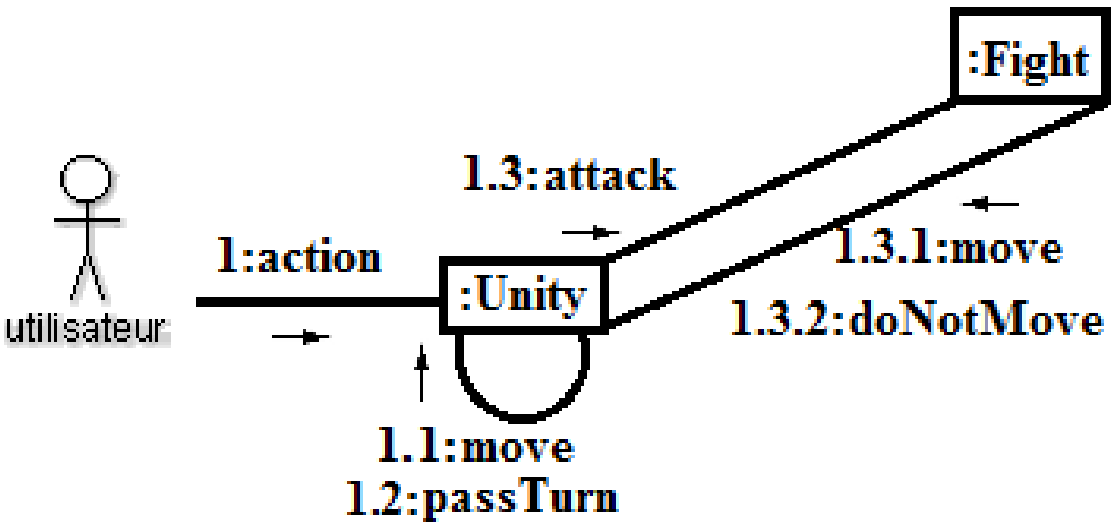

FIGURE 6 – Les combats d'une pièce par diagramme de communication

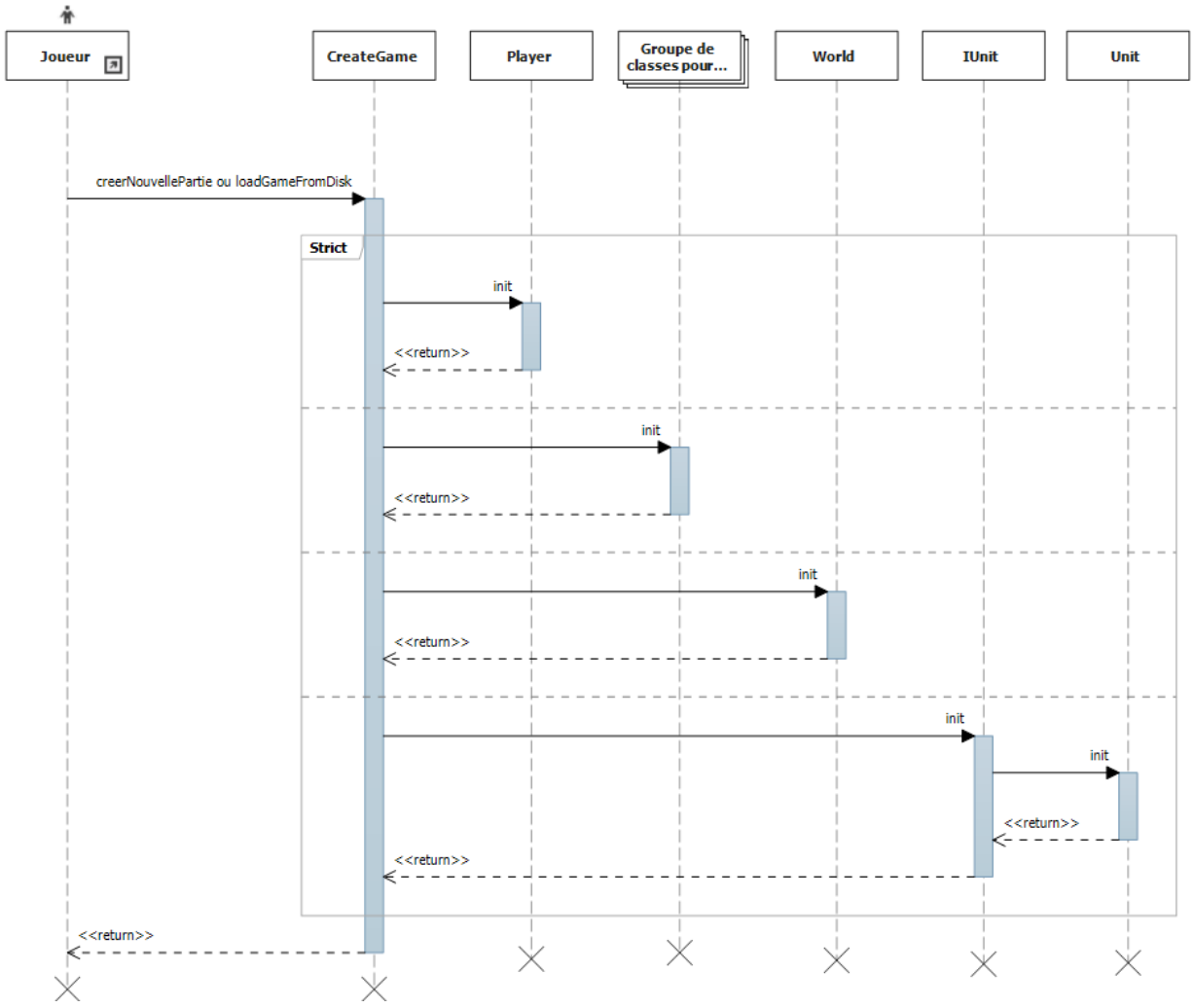

FIGURE 7 – Initialisation du jeu

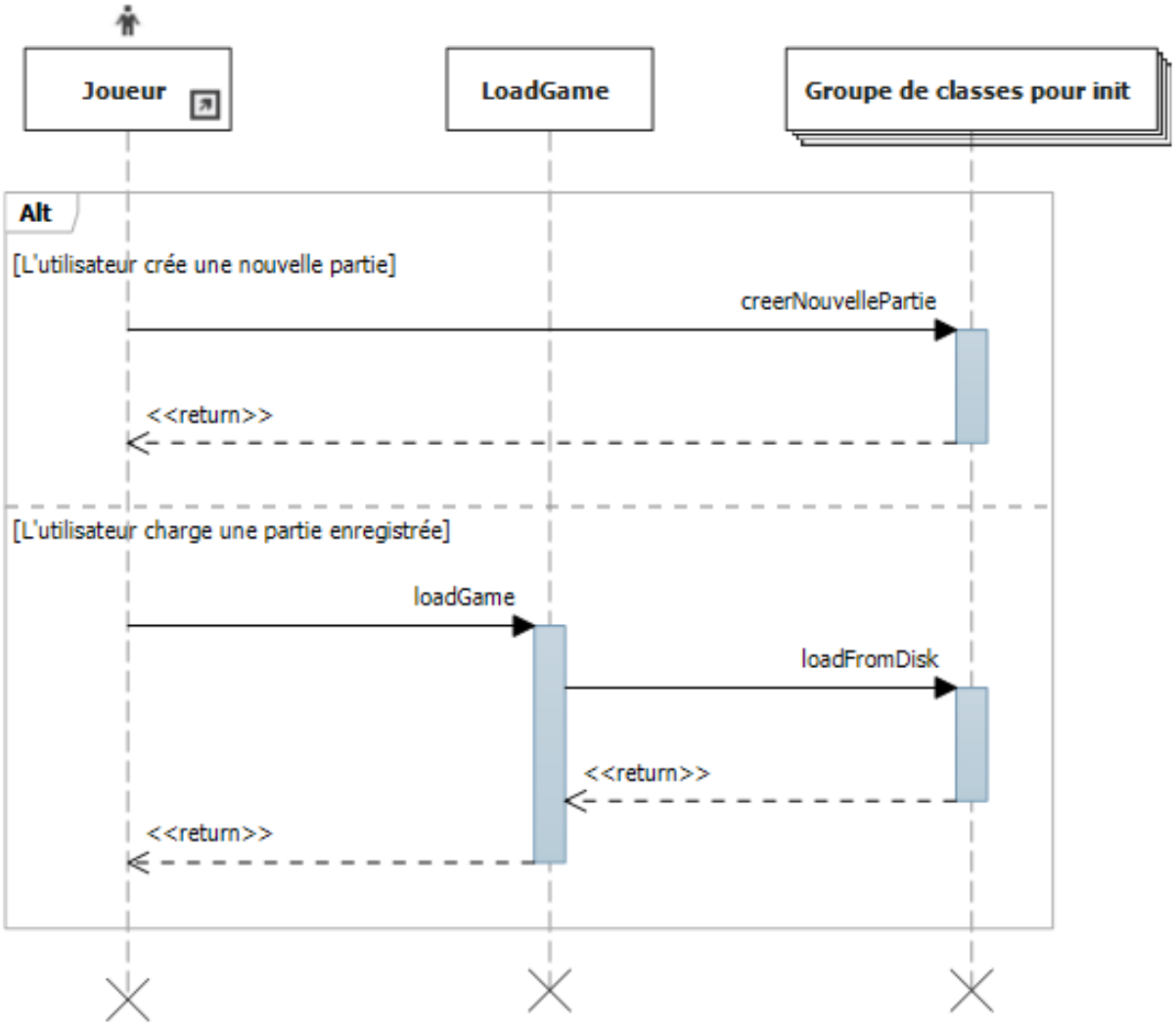

FIGURE 8 – Initialisation du jeu par diagramme de séquence

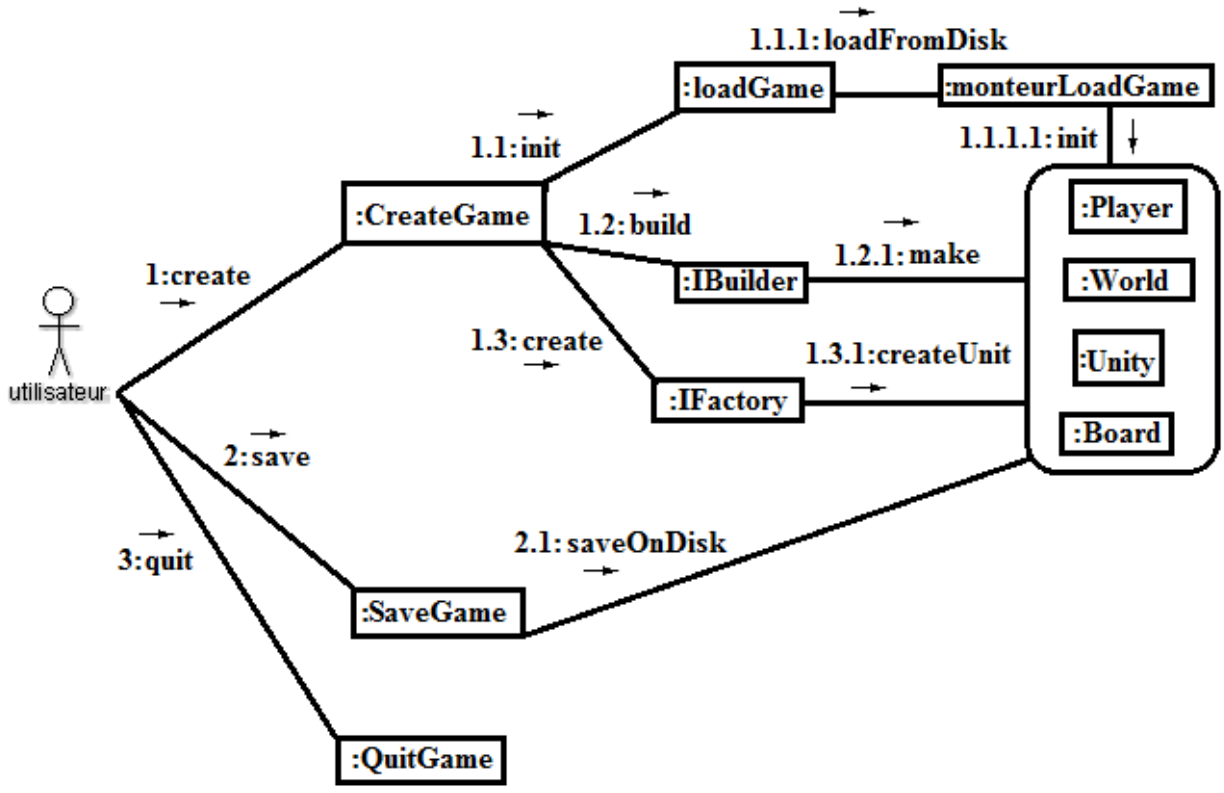

FIGURE 9 – Initialisation du jeu par diagramme de communication

4 Structure du code

5 Conclusion