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

Notre projet consiste à programmer un jeu de plateau entre Civilisation et Smallworld sur PC. Le but de ce projet est d'apprendre à programmer en C#, qui est un langage orienté objet, tout en utilisant des méthodes de conception telles que les patrons de conception ou la création de plusieurs diagrammes. Dans ce but, nous avons conçu des diagrammes de classe, de cas d'utilisation, d'intéractions, d'états-transition qui aident à visualiser le fonctionnement du jeu. Nous allons dans un premier temps survoler le principe du jeu, pour ensuite étudier en détail ses règles. Enfin, nous détaillerons la structure globale du code à l'aide de diagrammes de classes.

2 Généralités sur le jeu

Le but du jeu est d'avoir plus de points que l'adversaire au bout de 5/20/30 tours (selon la difficulté). Chaque joueur commence avec un certain nombre d'unités (également selon la difficulté) qu'il pourra déplacer à chaque tour afin d'occuper plus de cases ou de détruire des unités adverses. Le placement de chaque unité rapporte plus ou moins de points.

2.1 Carte du monde

Les unités se déplaceront sur une carte de taille variable selon la difficulté. Les cases seront hexagonnales 2.1, les unités auront donc au maximum 6 déplacements/attaques possibles. IL existe trois type de tuile possible : plaine, forêt et montagne. Les différentes races aront des interactions spéciales avec ces tuiles.

FIGURE 1 – Cases de la carte

2.2 Lancement du jeu

Au lancement du jeu, le joueur peut soit créer une nouvelle partie, soit en charger une comme l'explique le DIAGRAMME 2.2. Lors de la création d'une nouvelle partie, le joueur peut personnaliser sa partie de différentes façons, lui permettant d'avoir des éxpériences différentes à chaque partie. Il peut tout d'abord choisir la difficulté, ce qui influence plusieurs paramètres :

- Démo : 6x6 cases ; 5 tours ; 4 unités par peuple
- Petite : 10x10 cases ; 20 tours ; 6 unités par peuple
- Normale : 14x14 cases ; 30 tours ; 8 unités par peuple

Chaque joueur peut ensuite choisir sa race parmi les 3 races disponibles : Elfe, orc et nain. Chaque race confère des bonus détaillés en partie 2 de ce rapport.

>include graphic cas d'utilisation nouvelle partie/charger

FIGURE 2 – Lancement du jeu

2.3 déroulement d'un tour

Nous allons maintenant expliciter le déroulement d'un tour. A chaque tour, le joueur peut déplacer chaque unité qu'il contrôle si celle-ci a des points de déplacement. Si une unité adverse est sur une case adjacente, il peut à la place attaquer cette unité pour tenter de prendre le contrôle de cette case. Le déroulement des combat est expliqué dans la partie 2 de ce rapport.

3 Approche en détail

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

Le diagramme d'état transition de la vue d'ensemble en Annexe A permet de situer les différents cas possibles de manière générale. Le détail sera fait dans les parties suivantes.

Le cycle de l'unité est le suivant : elle est créée, 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 unités 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

Le diagramme d'état transition de déplacement d'une pièce en Annexe B 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

Le diagramme d'état transition du combat d'une pièce en Annexe C 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 % = [0,35[), il peut infliger des blessures à son adversaire, et inversement. Ainsi, le perdant du tour de combat perd Y vies, Y étant la valeur d'attaque de l'unité pondéré par le pourcentage de vie de l'unité. Si l'unité compte 2 points de vie, sachant qu'elle a démarré avec 5 points de vie, alors l'attaque sera multipliée par 40%. Le compteur de nombre de combats est décrémenté et on regarde ensuite s'il reste encore des tours de combat. Si oui, on refait le calcul de probabilité, et ainsi de suite. Sinon, on stoppe le combat par un match nul. L'attaquant ne bouge pas. Son tour est terminé. Si durant le combat, une unité n'a plus de point de vie, elle est détruite.

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 50% de malchance de prendre des dégats. S'il a 3 att. contre 4 déf., le rapport de force est de 75% : 3/4 = 25%, 25% de 50% = 12.5%, 50%+12.5

3.1.4 Fin de combat

Le diagramme d'état transition Fin de Combat en Annexe D 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 sur le terrain Plaine. Dans tous les autres cas, toutes les unités gagnent un point de victoire. Il leur est ensuite possible de gagner un autre point de victoire, si l'unité est un est un orc placé sur une case Foret, 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, les points de victoire bonus disparaissent avec lui.

3.2 Interactions entre les différentes composantes du jeu

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

3.2.1 Initialisation

Le diagramme de séquence du chargement du jeu en Annexe E nous 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 le diagramme de communication de la création du jeu en Annexe F, on peut visualiser les appels effectuant cette initialisation.

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

Avec le diagramme de création du plateau Board en Annexe G, on peut comprendre l'ordre de création du Board, avec l'appel à une interface, qui appelera un monteur qui créera les cases pour l'AbstractBoard, qui initialisera les cases Tiles et leur Position associée.

3.2.2 Combats et déplacements

Les diagrammes de séquence et de communication respectivement en Annexes I et J présentent 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.

4 Structure du code

Nous allons maintenant faire une ébauche de la structure de notre code, avec les différents patrons de conception utilisés.

4.1 Patrons de conception

Pour répondre aux impératifs du projet, nous utiliserons cinq patrons de conception dans le but d'obtenir un code clair, optimisé et facile à entretenir.

- Fabrique : La création des unités de chaque joueur lors de leur placement en début de partie sera gérée par le patron "Fabrique"
- Poids-mouche : Afin d'éviter de dupliquer des objets identiques dans la mémoire tout en gardant un code simple, les tuiles de la carte ne seront instantiées qu'une seule fois par type (plaine, forêt et montagne), la carte ne contiendra que des liens vers ces instances grace au patron "Poids-mouche".
- Stratégie : Les différentes difficultés du jeu (taille de la carte, nombre de tours, nombre d'unités par joueur) seront implémentés via le patron "Stratégie"
- Monteur : La fabrication d'une carte lors d'une nouvelle partie ou d'un chargement sera mis en oeuvre par le patron "Monteur"
- Modèle-Vue-Contrôleur : Le code adoptera le patron "Modèle-Vue-Contrôleur", ce qui permet

FIGURE 3 – Diagramme de classe : patrons de conception

4.2 Structure du code

La structure de la partie "modèle" du code source est modelisée par ce DIAGRAMME 4.2. Il ne représente que la partie "Modèle" de l'architecture, les parties "Vue" et "Contrôleur" ne sont pas affichées.

FIGURE 4 – Diagramme de classe : structure globale

5 Conclusion

A faire

6 Annexes

6.1 Annexe A : Diagramme d'état transition de la vue d'ensemble

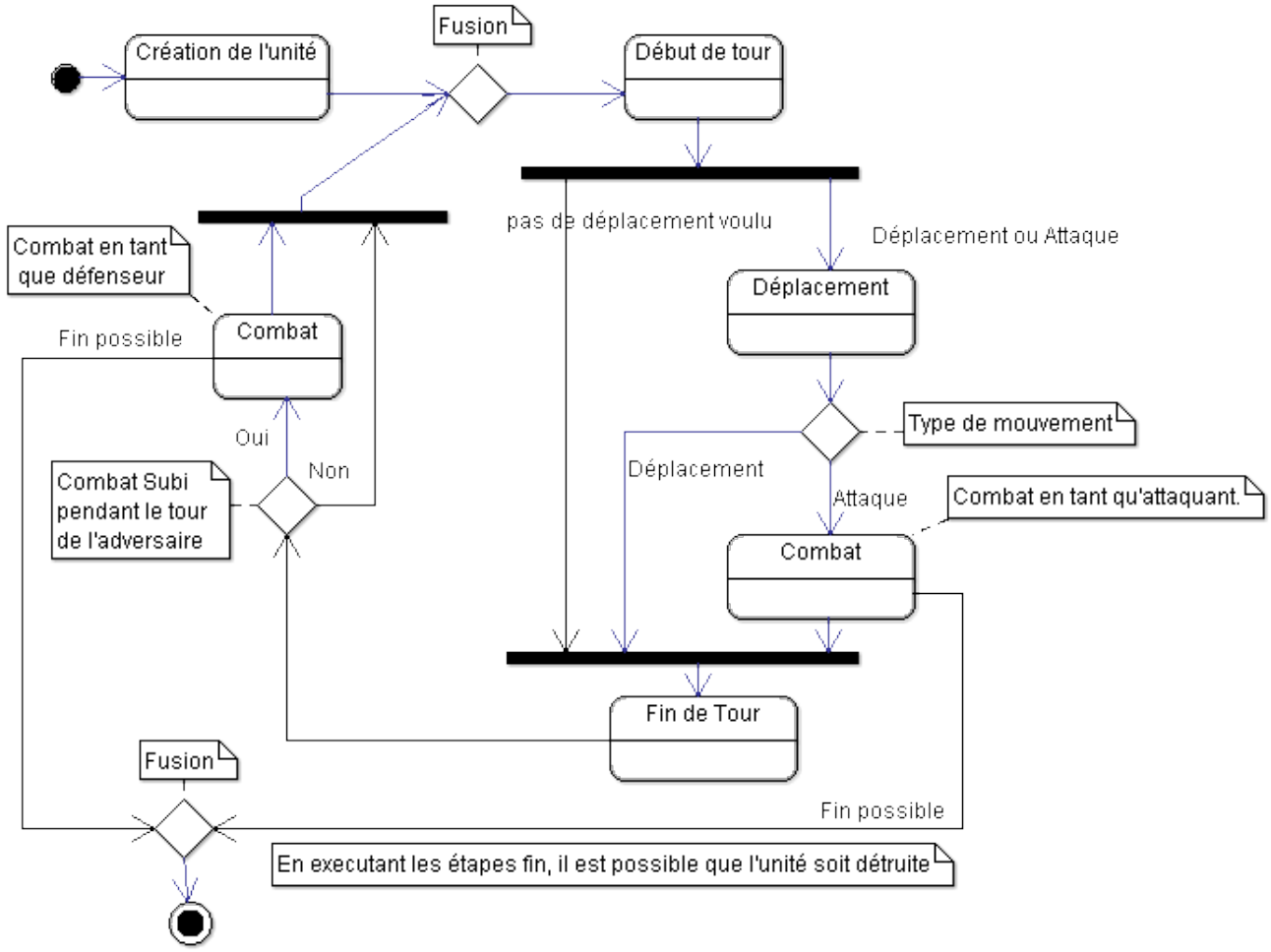

FIGURE 5 – Diagramme d'état transition de la vue d'ensemble

6.2 Annexe B : Diagramme du déplacement d'une pièce

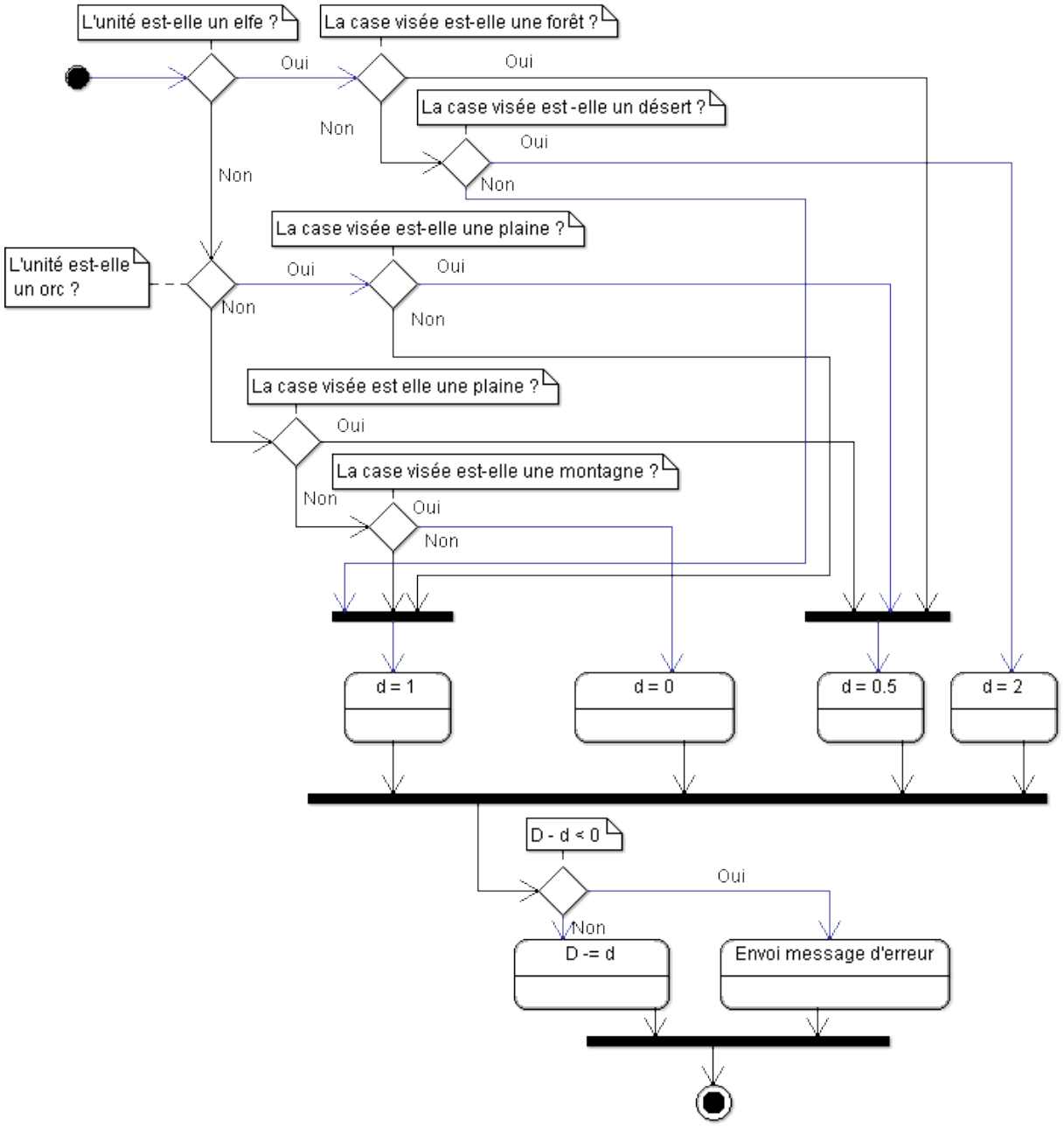

FIGURE 6 – Diagramme d'état transition du déplcement d'une pièce

6.3 Annexe C : Diagramme d'état transition du combat d'une pièce

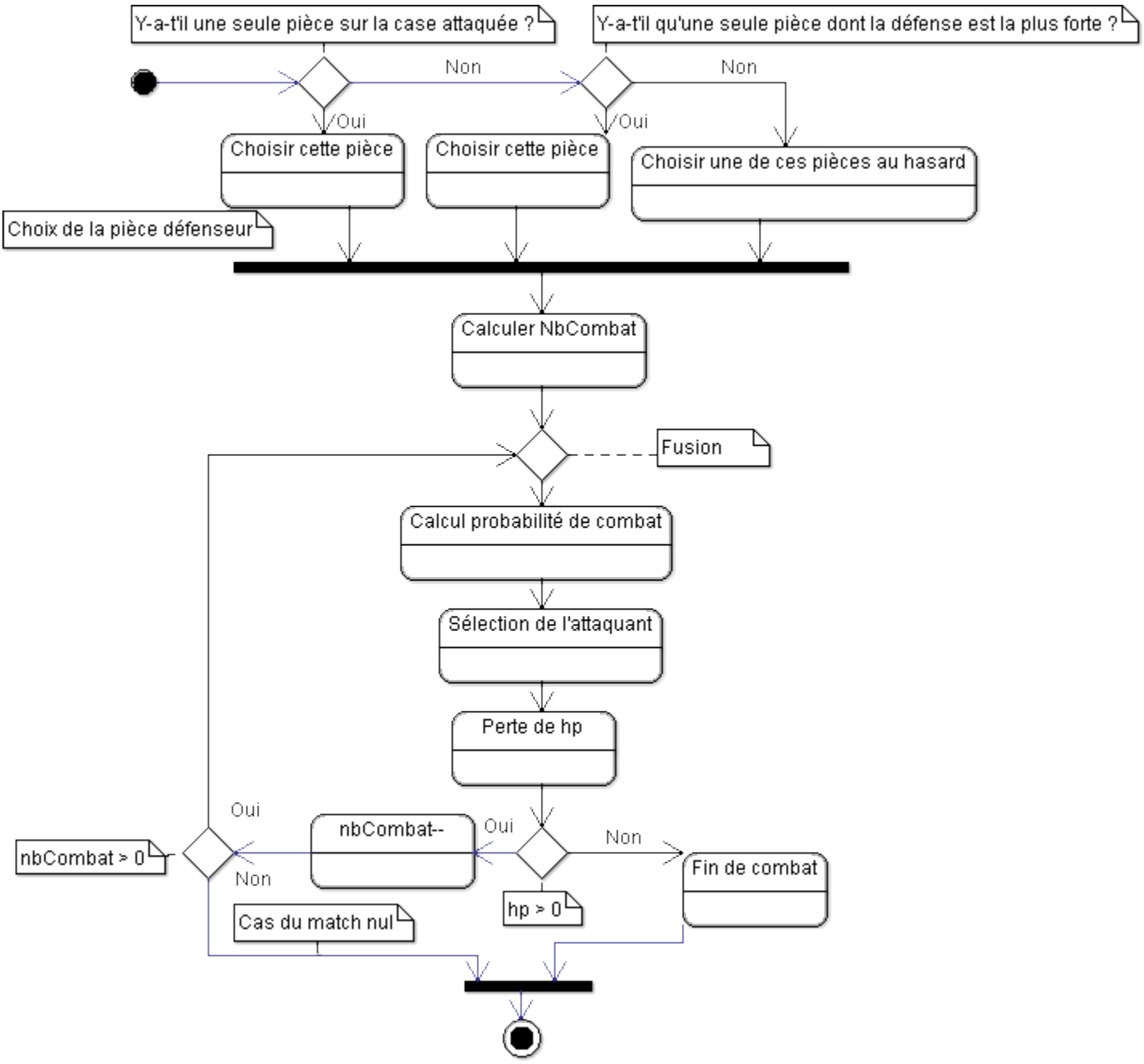

FIGURE 7 – Diagramme d'état transition du combat d'une pièce

6.4 Annexe D : Diagramme d'état transition des cas de fin de combat d'une pièce

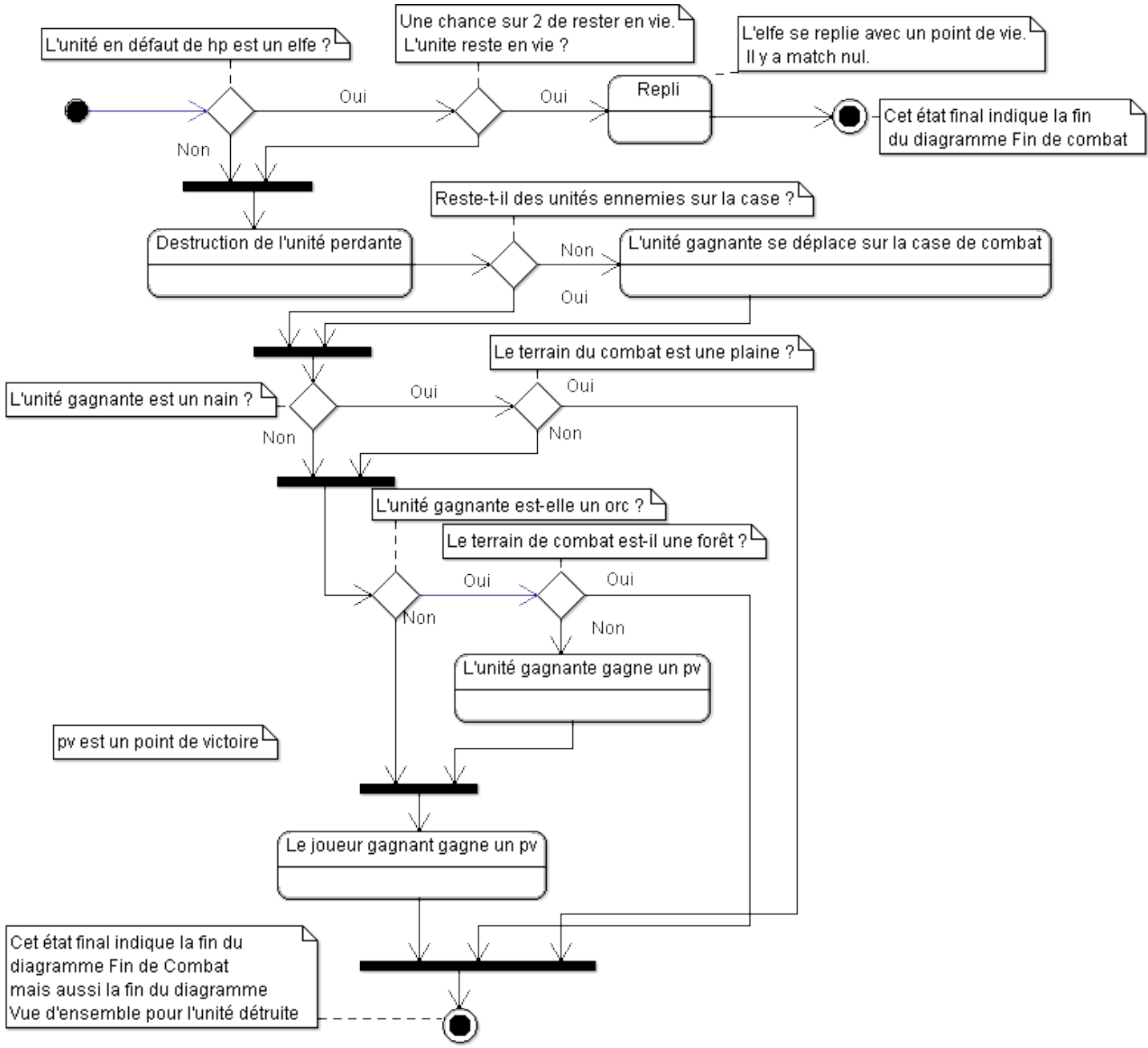

FIGURE 8 – Diagramme d'état transition des cas de fin de combat d'une pièce

6.5 Annexe E : Diagramme de séquence du chargement du jeu

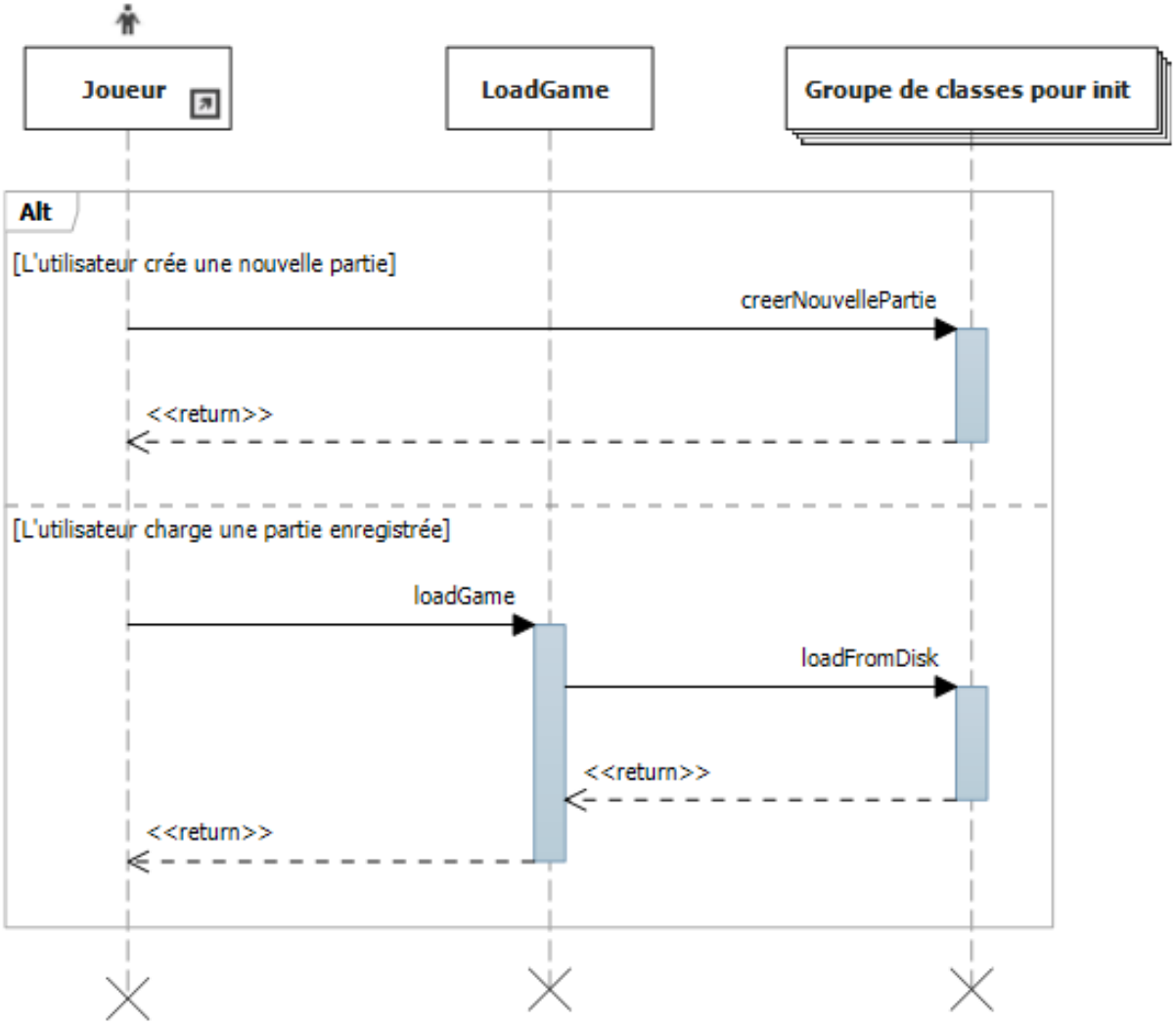

FIGURE 9 – Diagramme de séquence du chargement du jeu

6.6 Annexe F : Diagramme de communication des actions de l'utilisateur

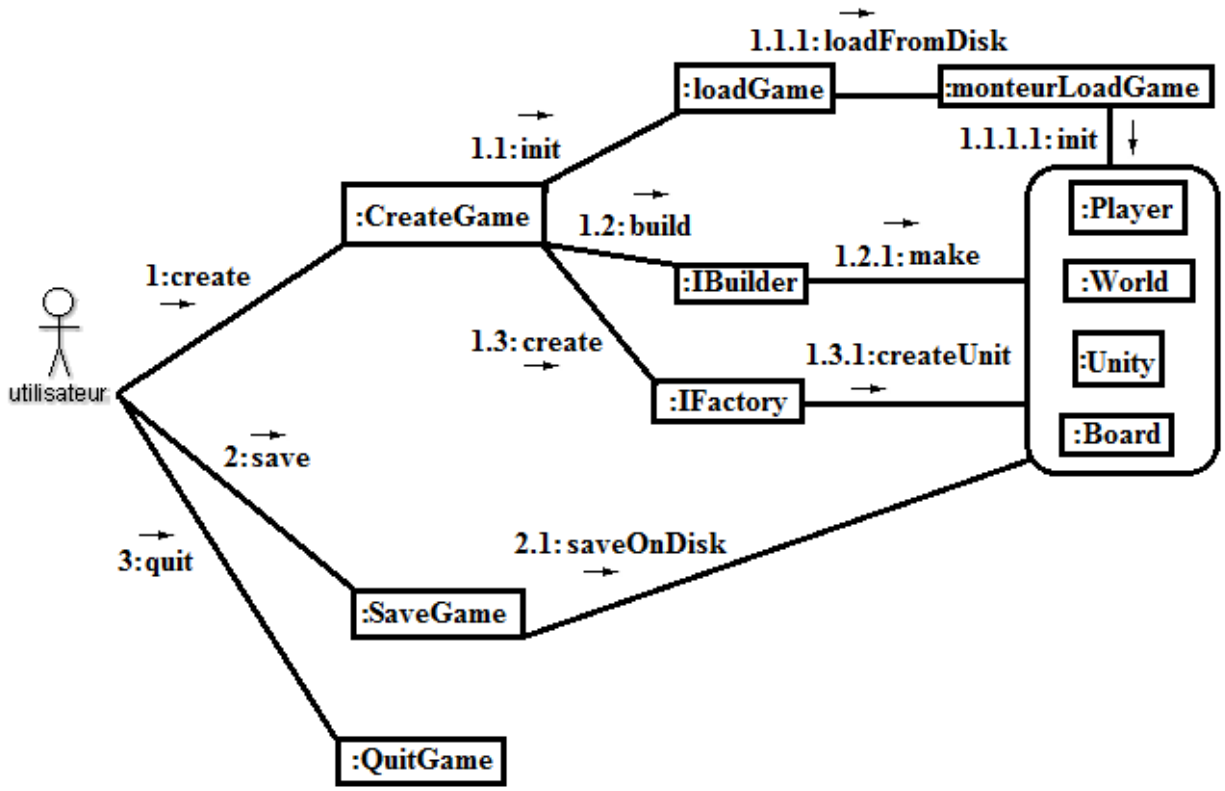

FIGURE 10 – Diagramme de communication des actions de l'utilisateur

6.7 Annexe G : Diagramme de séquence de l'initialisation globale

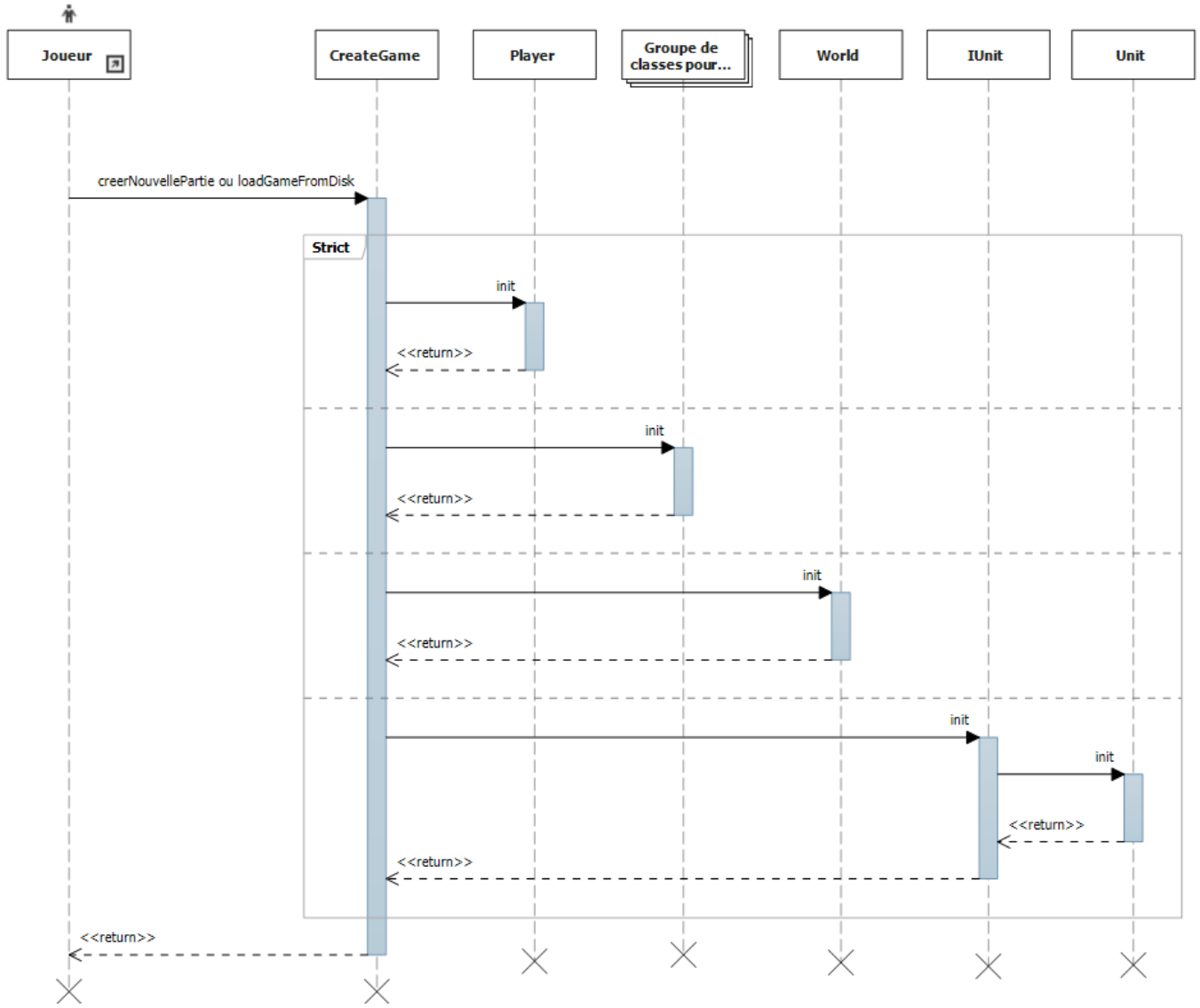

FIGURE 11 – Diagramme de séquence de l'initialisation globale

6.8 Annexe H : Diagramme de séquence de l'initialisation du board

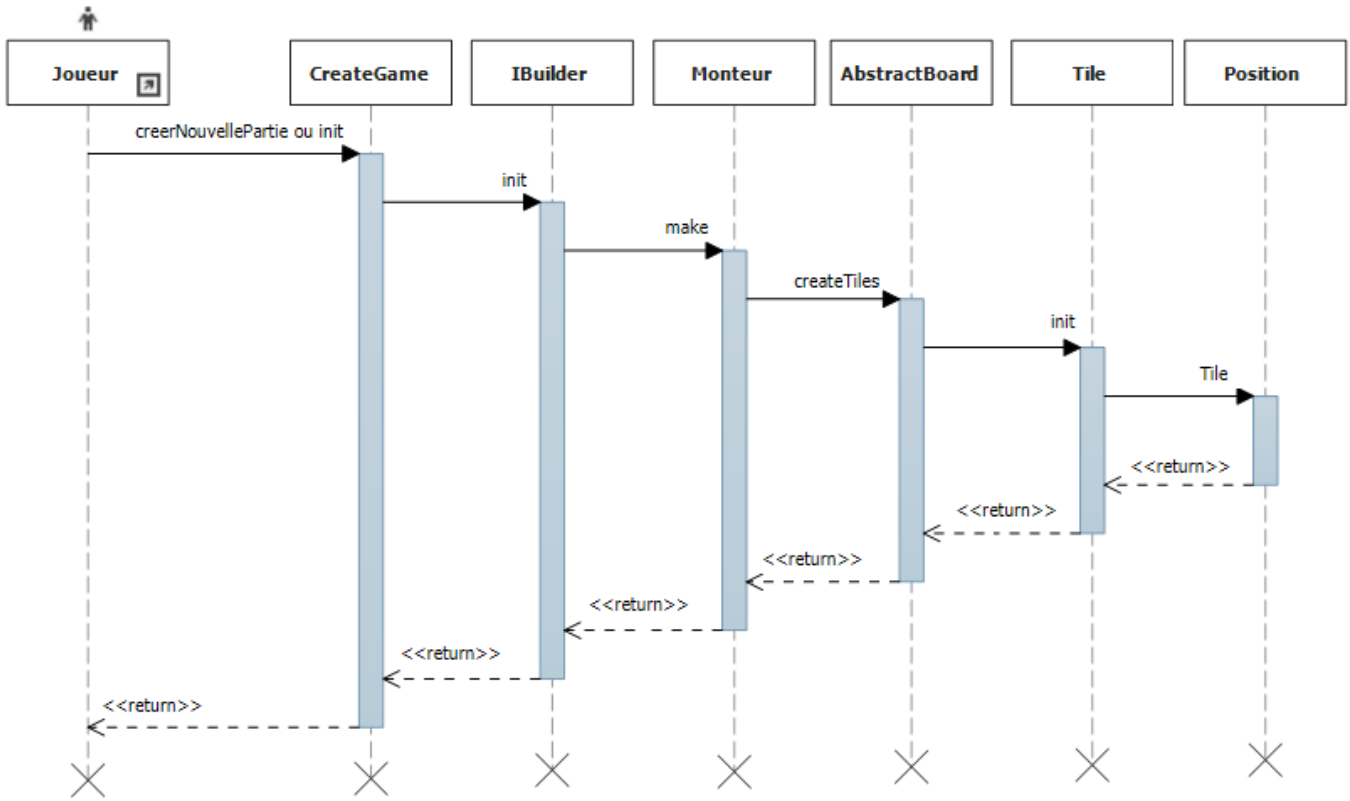

FIGURE 12 – Diagramme de séquence de l'initialisation du board

6.9 Annexe I : Diagramme de séquence des actions d'une pièce

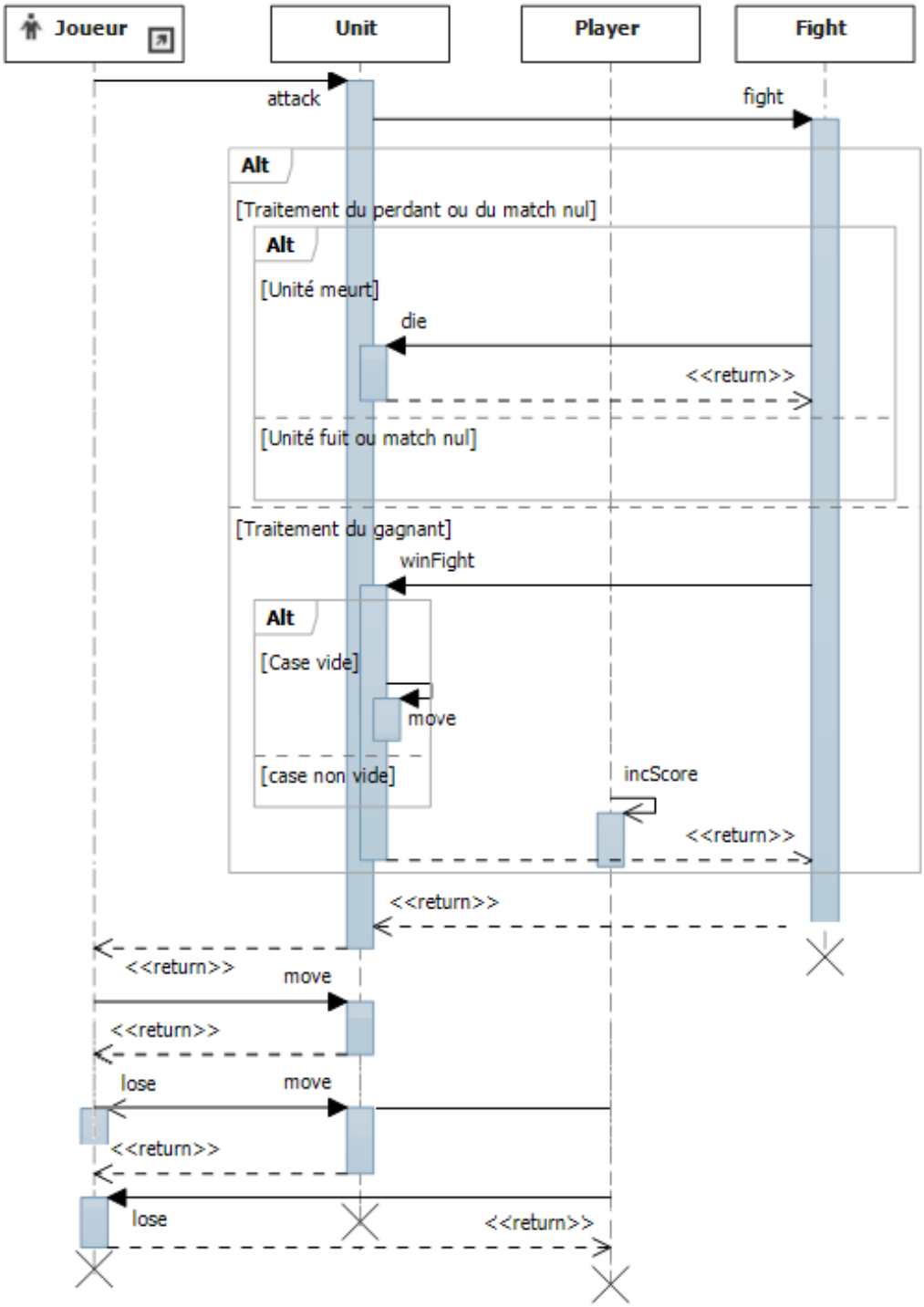

FIGURE 13 – Diagramme de séquence des actions d'une pièce

6.10 Annexe J : Diagramme de communication des actions d'une pièce

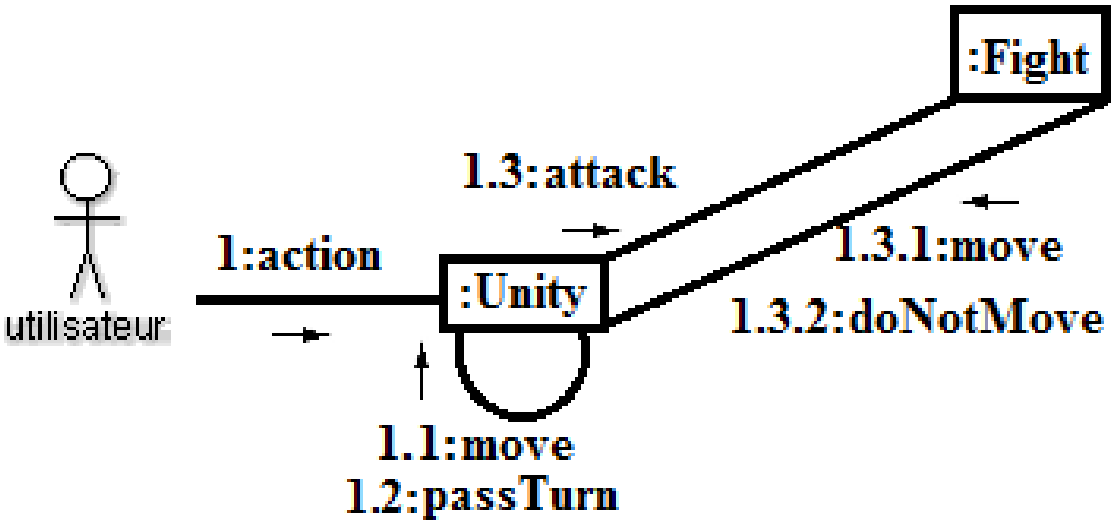

FIGURE 14 – Diagramme de communication des actions d'une pièce