Rapport de conception

Projet POO/MOO

4ème année

Maxime Lorant

Clarisse Renou

4INFO – 2013-2014

Table des matières

I. Introduction

Le but de ce projet, inscrit dans le cadre du cours de POO/MOO de 4ème année, est de **développer un jeu de tour par tour** inspiré de SmallWorld. Ce projet permet de mettre en pratique les compétences obtenus en C++, C# et en modélisation au cours du semestre.

Le jeu se jouera en local, sur la même machine, à 2 joueurs uniquement. Un joueur doit gérer un peuple et obtenir le plus de points à la fin de la partie, en plaçant judicieusement ses unités ou en éliminant celles de l'adversaire. Après l'initialisation de la partie, le jeu se déroule sur une carte du monde et affiche quelques informations sur le statut de la partie en dessous. Une maquette de l'interface est visible ci-dessous, en Figure 1.

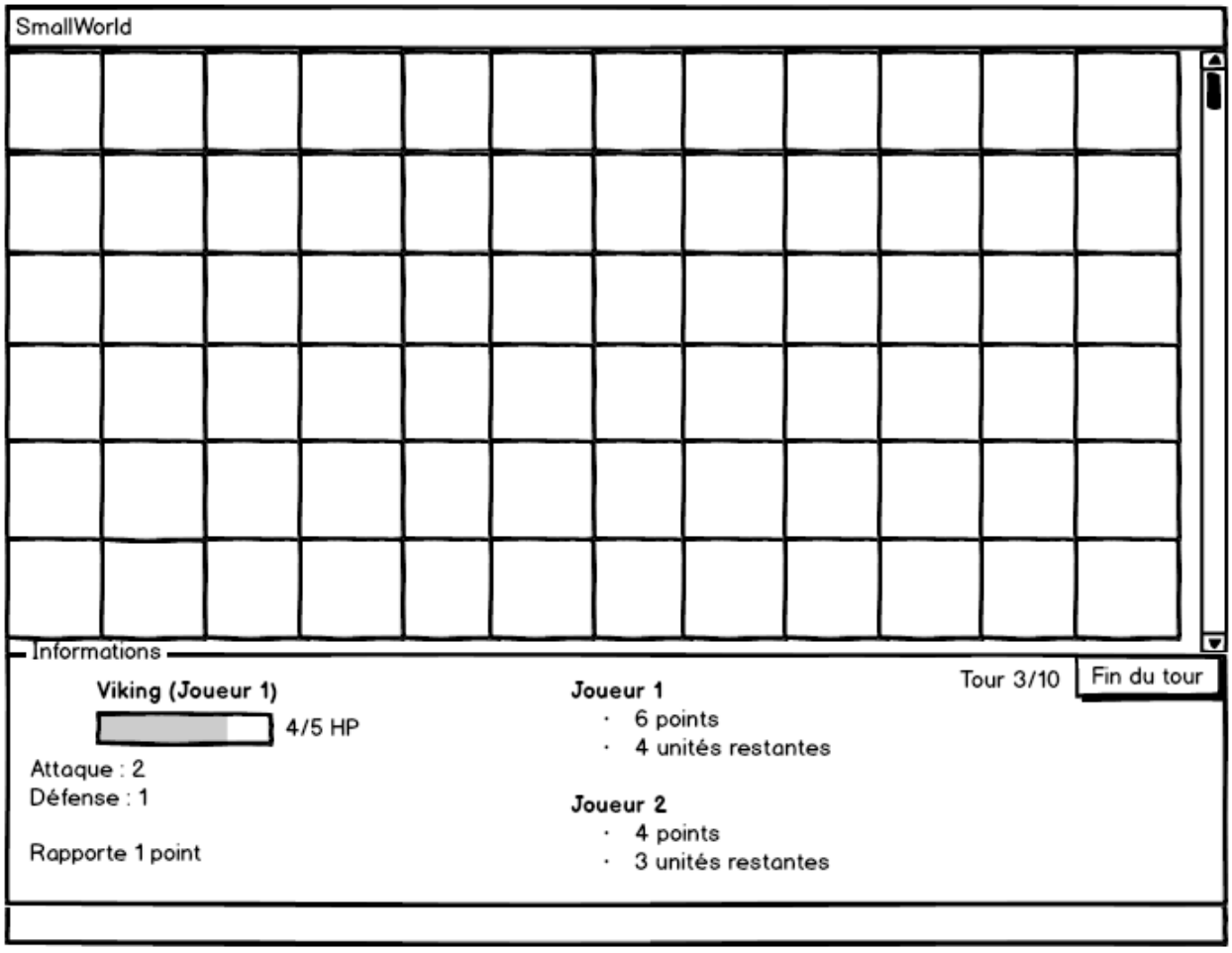

Figure 1 : Maquette de l'interface du jeu

II. Cas d'utilisation

Le jeu propose les cas d'utilisation suivant à l'utilisateur :

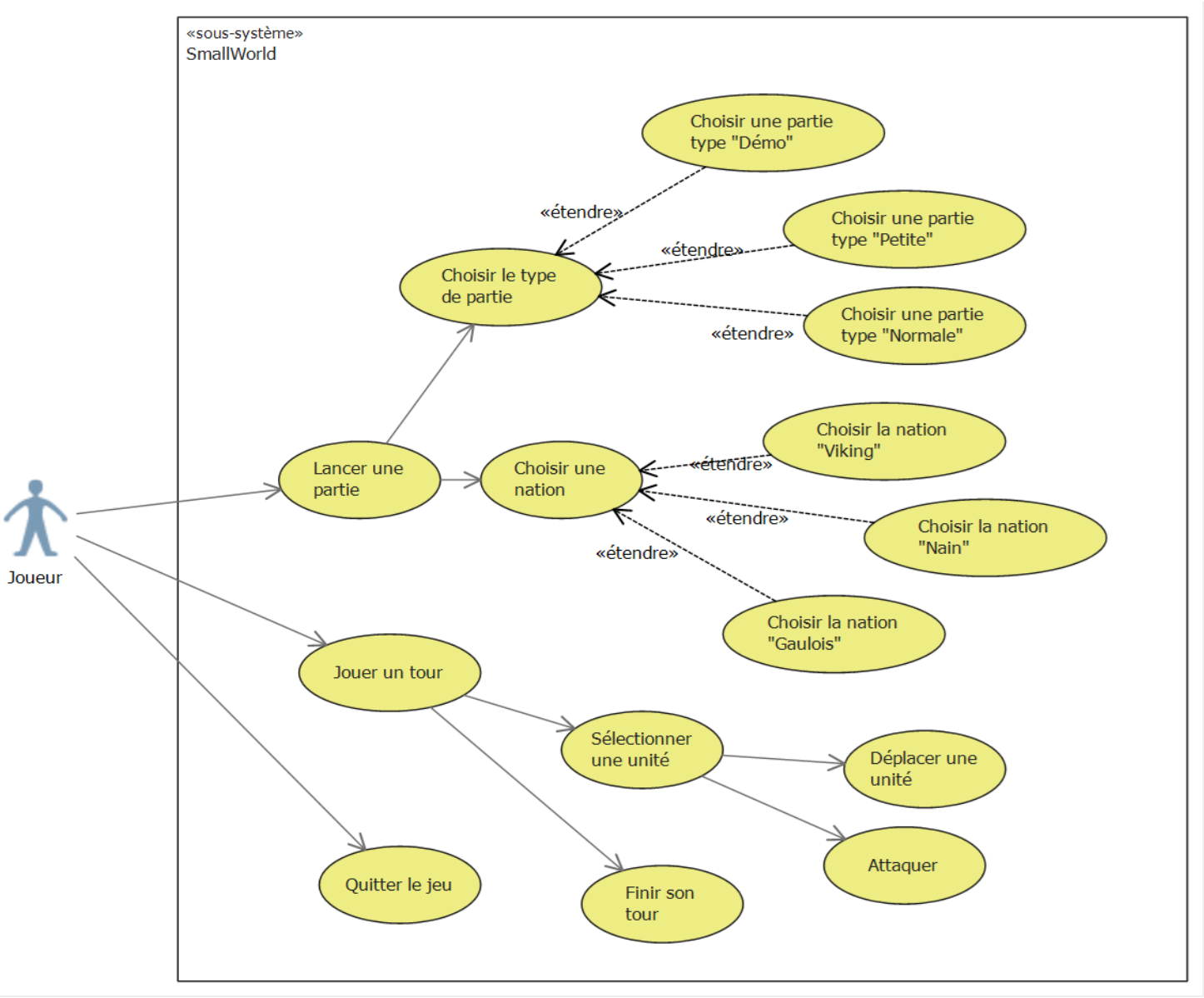

Figure 2 : Diagramme de cas d'utilisation

Il n'y a qu'un type d'acteur dans notre application qui est un joueur (représentant le joueur 1 ou le joueur 2). Le joueur peut donc effectuer toutes les actions possibles dans le jeu. Pour chaque cas d'utilisation décrit ci-dessous, l'acteur impliqué est donc forcément un joueur.

Lancer une partie

Permet à l'utilisateur de créer une nouvelle partie en choisissant le type de la partie. Ce cas se divise en 3 actions possibles : choisir une partie de type « Démo », « Petite » ou « Normale », ce qui influe sur la taille de la carte, le nombre d'unités et de tours de la partie.

Les joueurs pourront ensuite choisir leurs nations, parmi les « Viking », « Gaulois » et « Nain ». Le jeu sera initialisé avec une nouvelle carte, avec un placement des unités le plus éloigné possible des 2 adversaires.

Jouer un tour

Après avoir lancé une partie, les joueurs pourront jouer chacun à leur tour. Lors de leur tour, ils pourront sélectionner une unité et soit la déplacer sur une autre case, soit attaquer avec. Une fois les points d'actions de l'unité à 0, il ne pourra plus l'utiliser. Quand il a utilisé toutes ses unités, ou qu'il souhaite passer son tour, le joueur peut alors « Finir son tour », via le bouton correspondant (« Fin du tour » sur la Figure 1).

Quitter la partie

Il est possible d'arrêter le jeu en cours. Cette action a pour effet d'arrêter la partie en cours et de quitter le jeu. Aucun système de sauvegarde n'est prévu pour reprendre une partie précédemment interrompue.

III. Diagramme de classe

Le diagramme de classe représente la modélisation globale du jeu et les liens qu'il y a entre la carte, les joueurs et leurs unités. (cf Figure 3, page 7). Le diagramme d'interface associé est également disponible, fournie en annexe.

L'ensemble d'une partie se déroule autour de la classe `Game`, qui lie tous les éléments entre eux. C'est cette classe qui va permettre de diriger le déroulement du jeu, notamment à la création de la partie puis dans le déroulement des tours et enfin déterminer le gagnant. Ensuite, chaque donnée du jeu (joueur, carte, unités) sont représentées par des classes indépendantes. La plupart des composantes utilise un patron de conceptions parmi les suivants : le poids-mouche, la fabrique et stratégie.

Poids-mouche

Les cartes du jeu peuvent faire de 25 à 225 cases, en fonction du type de partie. Le patron de conception « Poids-mouche » permet d'éviter d'avoir un nombre important de cases à instancier. Nous aurons au maximum 5 objets de type `Case` instanciés, qui correspondront à chaque type de case qu'il peut y avoir dans le jeu : désert, plaine, eau, montagne et forêt.

Ce choix nous oblige cependant à déporter certaines informations. Les unités seront placées dans un tableau `units` dans l'objet `Map` du jeu, afin de savoir quelles unités sont sur quelles cases. De même, la carte sera représentée par une grille (nommé `grid` sur le diagramme), où chaque case est une instance du type de case correspondant.

Fabrique

Lors de la création de la partie, un joueur doit choisir sa nation. Le choix de la nation va alors entrainer la création d'unités différentes. Afin de rendre le comportement de création d'unités générique, une fabrique est utilisée ici : il permet par la suite de créer les 5 unités du peuple, sans se soucier de son type (qu'il soit Nain, Gaulois ou Viking).

Stratégie

A la création de la partie, certains paramètres diffèrent selon le type de partie choisi : la taille de la carte, le nombre d'unités par joueur et le nombre de tours maximum. Pour ce faire, on utilisera ici le pattern Stratégie : lors du choix du type de partie, on instancie un objet de type `DemoGame`, `SmallGame` ou `NormalGame`, afin de garder en mémoire tous les paramètres dépendant du type de partie.

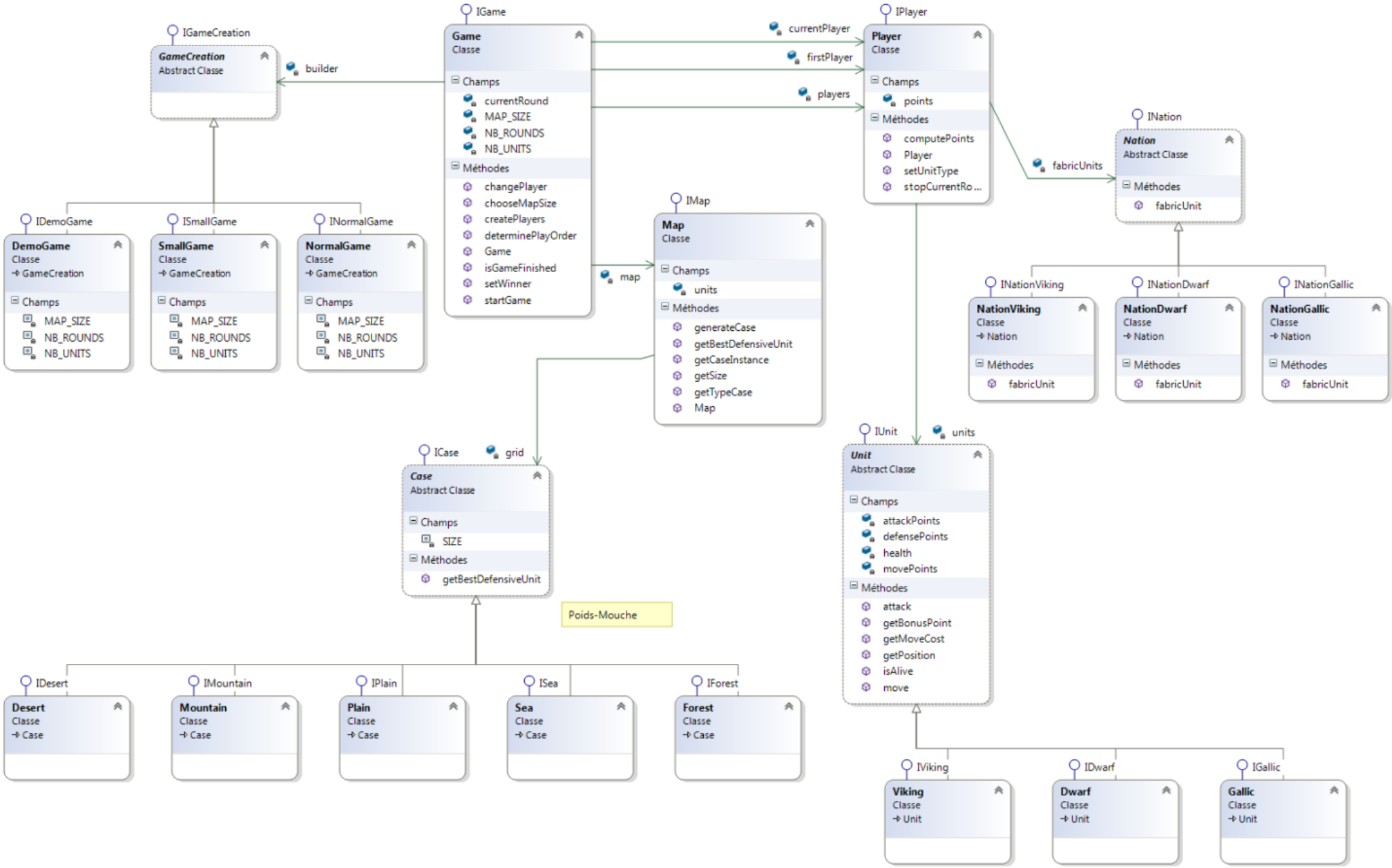

Figure 3 : Diagramme de classe globale du SmallWorld

IV. Diagrammes de séquence

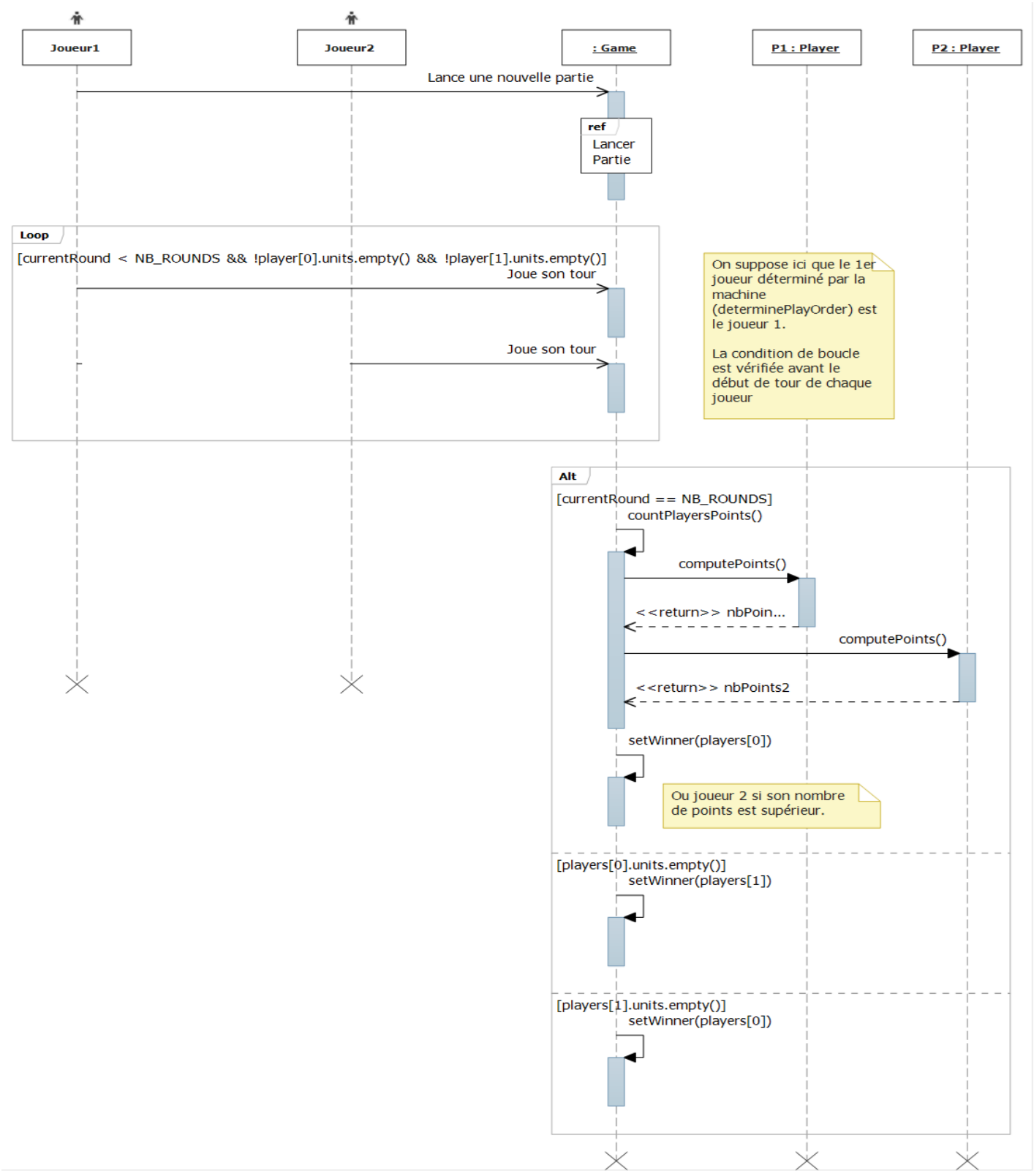

Figure 4 : Déroulement global d'une partie

Le diagramme de séquence Figure 4 représente le fonctionnement global du jeu, du lancement du jeu à la victoire d'un des joueurs. Une nouvelle partie est lancée lorsque le joueur décide de démarrer l'application. On pourra par la suite implémenter une fonction pour lancer une nouvelle partie sans relancer le jeu (cette fonction n'est pas spécifiée pour le moment).

Au lancement du jeu, le joueur doit donc créer une partie. Ce sous-système est développé plus bas, dans un second diagramme de séquence. Une fois la partie initialisée, les joueurs jouent alors chacun leur tour. De la même façon, le déroulement d'un tour est détaillé plus bas, dans un autre diagramme.

Une fois le jeu lancé, les joueurs finissent la partie si le nombre de tours maximum (défini à l'initialisation de la partie) est atteint, ou si l'un des deux joueurs n'a plus d'unités. Dans le premier cas, on compte les points (prenant en compte les caractéristiques de chaque unité et leur lieu sur la carte) et le vainqueur est celui qui a le score le plus important. Sinon, c'est le joueur qui a encore des unités en vie qui a gagné.

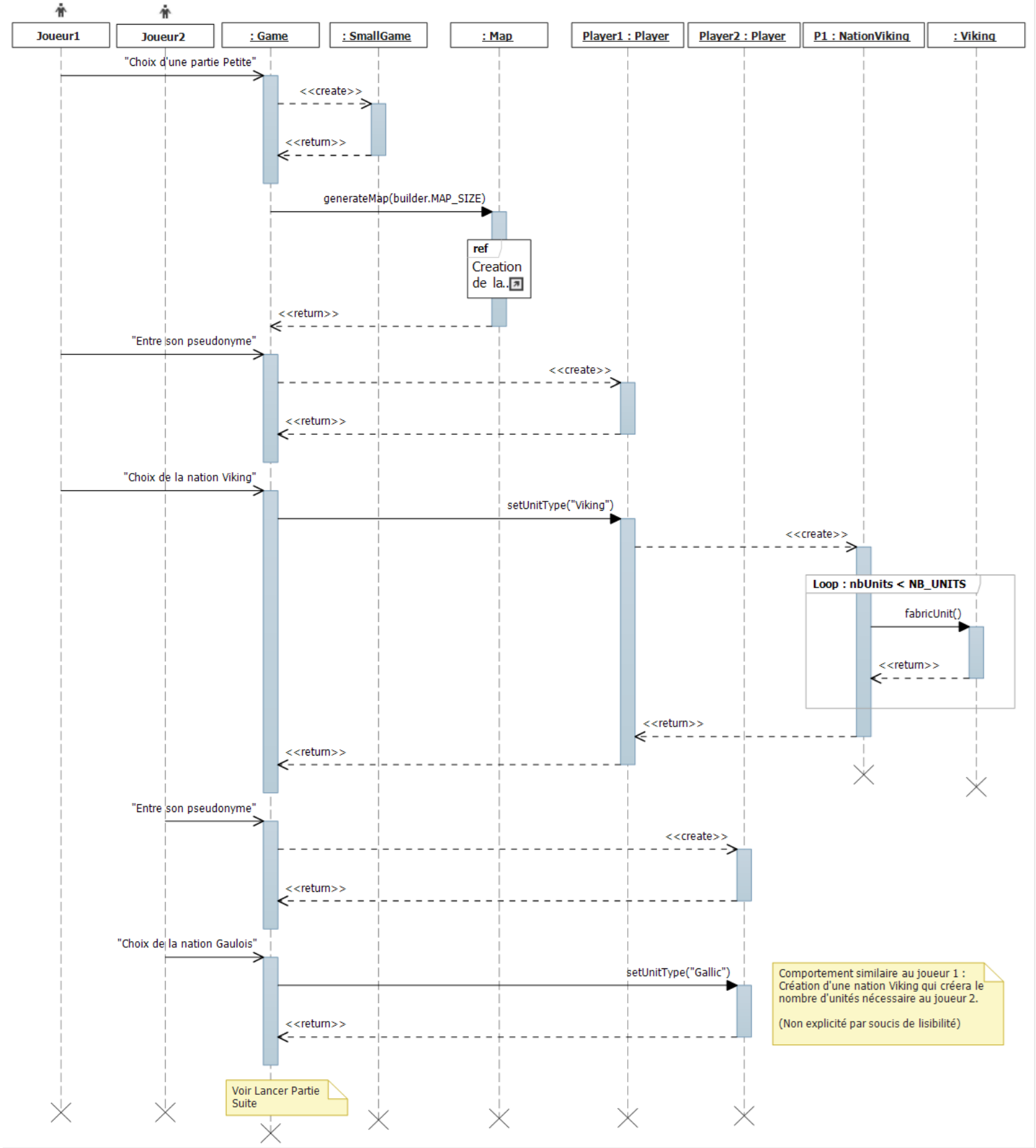

Figure 5 Diagramme de séquence : Initialisation d'une partie

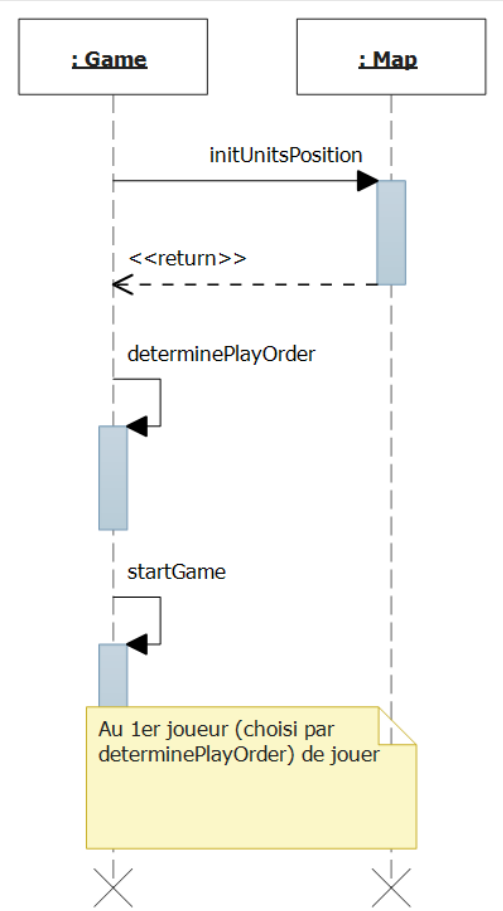

Figure 6 : Fin du lancement d'une partie (séparé pour plus de lisibilité)

Le diagramme de séquence de la Figure 5 et 6 présentent l'initialisation d'une partie. Tout d'abord un joueur sélectionne le type de partie qu'ils joueront : démo, petite et normale. Le jeu génère alors une carte aléatoire. Le premier joueur entre un pseudonyme et le `Player` est créé, il permettra d'accéder aux unités et fait le lien avec la carte et le jeu.

Ensuite, le jeu l'invite à choisir une nation, entre Gaulois, Viking et Nains. Selon le type de partie, 4, 6 ou 8 unités sont créées. C'est au joueur 2 de rentrer son pseudonyme, et les mêmes étapes s'enchainent sachant que deux joueurs ne peuvent sélectionner la même nation.

Le jeu place aléatoirement les unités mais de façon à séparer les adversaires le plus possible sur la carte. Le jeu commence alors, et c'est au joueur choisi par la méthode `determinePlayOrder` de jouer en premier.

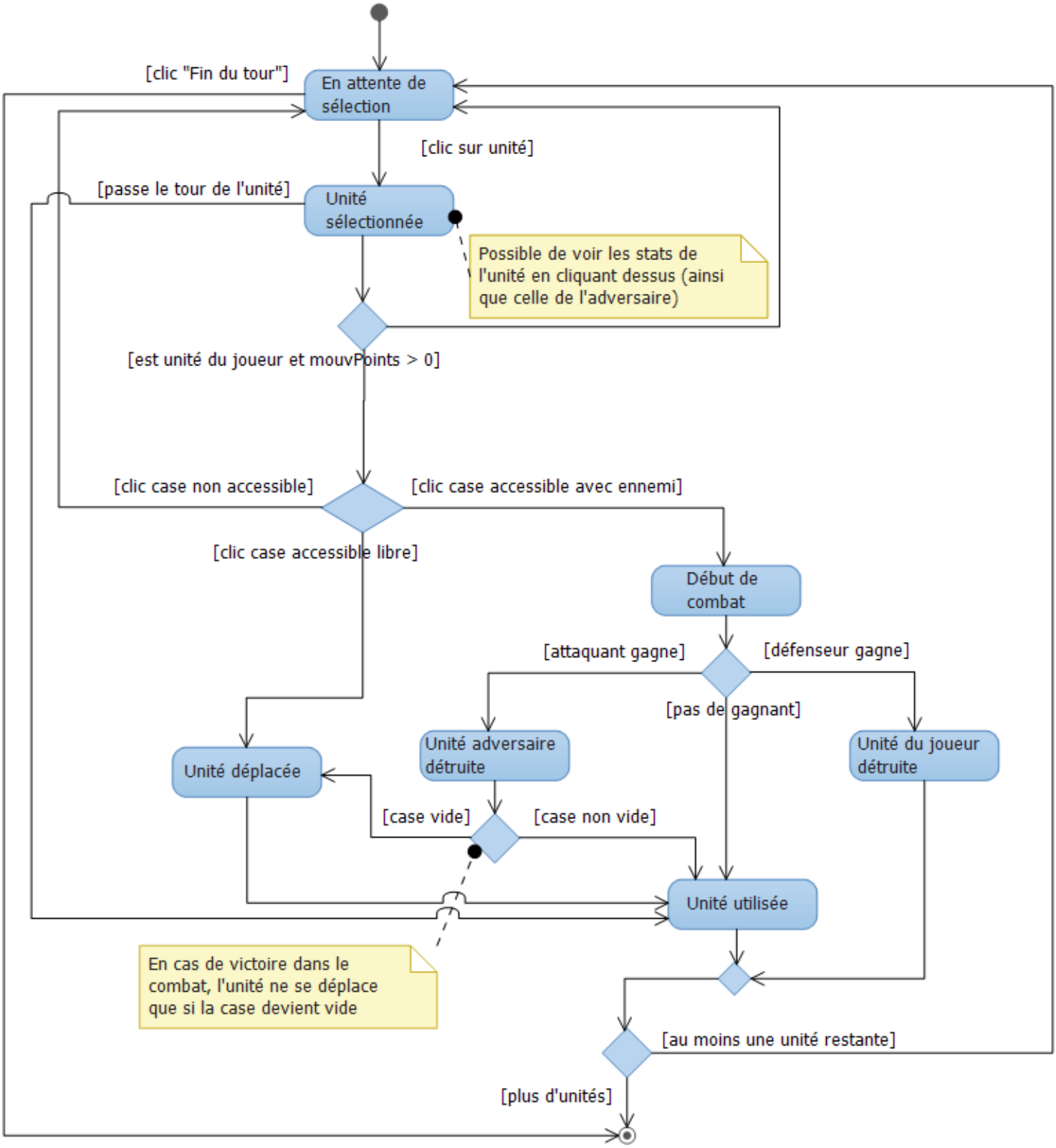

Figure 7 - Diagramme d'activité : déroulement du tour d'un joueur

La Figure 7 est un diagramme d'activité illustrant **l'algorithme du déroulement d'un tour** d'un joueur. Dès le début du tour, le joueur est en attente de sélection d'une entité. Le joueur a le choix entre sélectionner une unité ou passer son tour. Pour sélectionner une unité il clique dessus, les statistiques de l'unité s'afficheront tels que les points de vies, de mouvement restants etc. Le joueur peut également cliquer sur une unité adverse afin de voir ses statistiques mais aucune action n'est disponible dans ce cas.

Si l'unité lui appartient, il peut passer le tour de l'unité définitivement ou bien cliquer sur une autre unité pour y revenir plus tard et faire ensuite une autre sélection. En outre, il peut déplacer l'unité s'il lui reste des points de mouvements en cliquant sur une case. Si l'unité n'a pas assez de point de déplacement pour l'atteindre, l'unité ne bouge pas et un message éventuel affiche une erreur. Dans le cas d'une case est accessible, deux autres cas se présentent.

- Si elle ne contient pas d'unités ennemies, l'unité s'y déplace, les points de mouvements sont décrémentés et le tour de l'unité est terminé.
- Sinon, un combat débute contre la meilleure unité défensive de l'adversaire. En cas de victoire l'unité se déplace sur la case si elle est devenue vide et l'unité adverse est détruite, c'est la fin du tour de cette unité. En cas de défaite, l'unité du joueur est détruite. Sinon les 2 unités restent sur leurs cases respectives.

Une fois le tour de l'unité terminé, le joueur peut en sélectionner d'autres et les jouer s'il leur reste des points de mouvements, sinon c'est la fin du tour du joueur. Par ailleurs, le joueur peut passer son tour et donner la main au joueur adverse.

V. Diagramme d'états transitions

Afin de clarifier l'état des unités au cours du jeu, nous avons modélisé le cycle de vie d'une unité lors du jeu, de sa création jusqu'à sa mort :

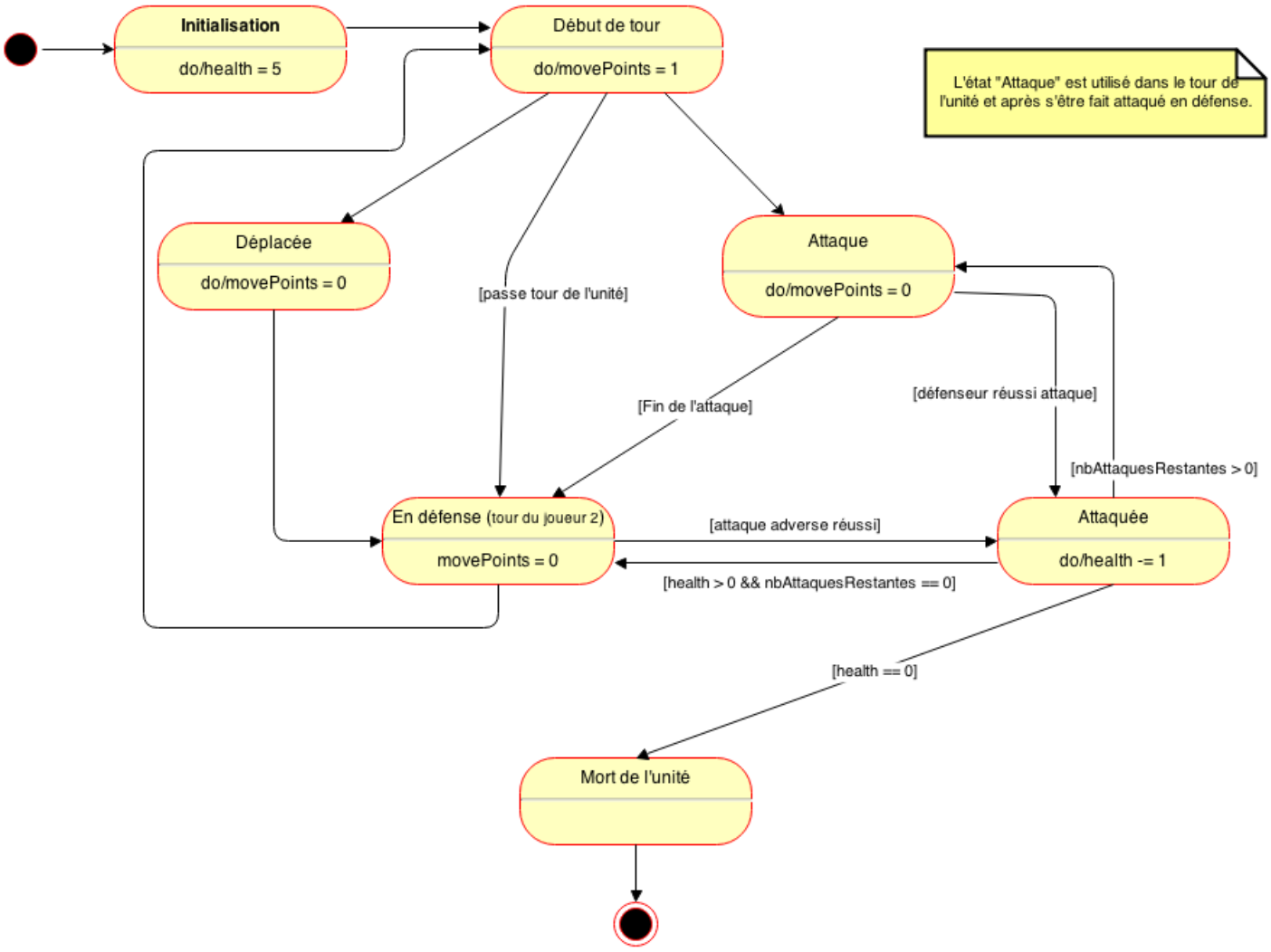

Figure 8 : Cycle de vie d'une unité

Toutes les unités ont 5 points de vie au début de la partie. A chaque début de tour du joueur, son compteur de déplacement est remis à 1, et peut être utilisé pour se déplacer ou attaquer. Une fois l'unité utilisée lors du tour (ou si elle est passée), elle passe dans l'état « défensif » où elle s'apprête à être attaquée par une unité ennemie.

Dans le cas d'une attaque, elle peut alors riposter ou se prendre des coups, faisant perdre 1 point de vie à chaque fois. Les probabilités d'attaque et de défense sont explicitées dans le sujet et dépendent des points d'attaque et de défense de chaque unité.