Rapport de conception SmallWorld

Case d'utilisation de création d'une partie

Dans le jeu SmallWorld, le Maitre du Jeu a la possibilité de créer une partie selon le type qui lui convient : une partie très courte (type Démo), Petite ou Normale. Cela impacte la taille de la carte, le nombre d'unité disponibles ainsi que le nombre de tour à jouer. De plus, les deux Joueurs peuvent avoir la possibilité de choisir le peuple qu'ils désirent. Ces réflexions se retrouvent dans le diagramme des cas d'utilisation ci-dessous (Illustration 1).

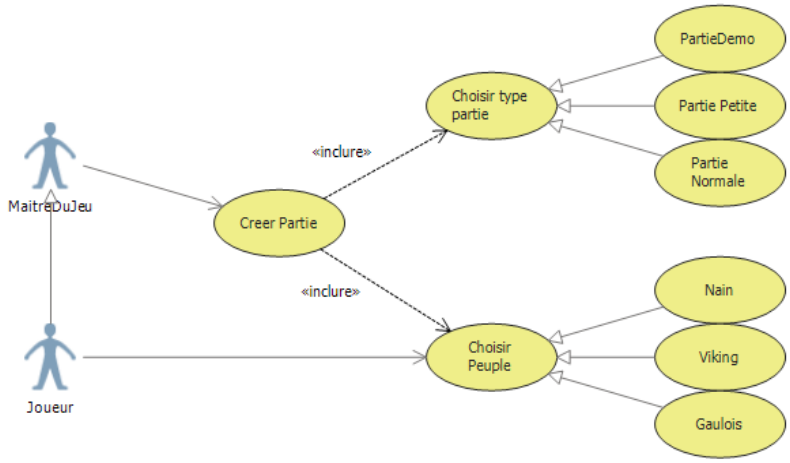

Illustration 1: Diagramme des cas d'utilisation de création d'une partie

Cas d'utilisations d'une unité

Le projet est basé sur le vrai jeu de plateau SmallWorld, se déroulant au tour par tour. À chaque tour, le joueur pourra utiliser ses unités de la manière dont il veut : attaquer, se déplacer ou ne rien faire. Afin de choisir au mieux ses déplacement, il doit aussi pouvoir analyser les cases de la carte. Tout cela est expliqué dans le diagramme des cas d'utilisation ci-dessous (Illustration 2).

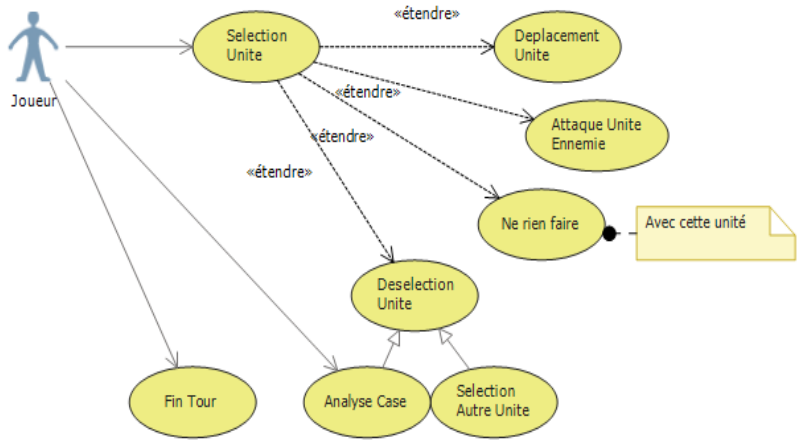

Illustration 2: Diagramme d'utilisation d'unité

Gestion des combats

L'algorithme des combats étant plus compliqué qu'un simple « si … alors … sinon », il a été nécessaire de réaliser un premier diagramme d'activité (Illustration 3) afin d'exprimer clairement l'algorithme général, ainsi qu'un deuxième (Illustration 4) afin pour celui des calculs de probabilité.

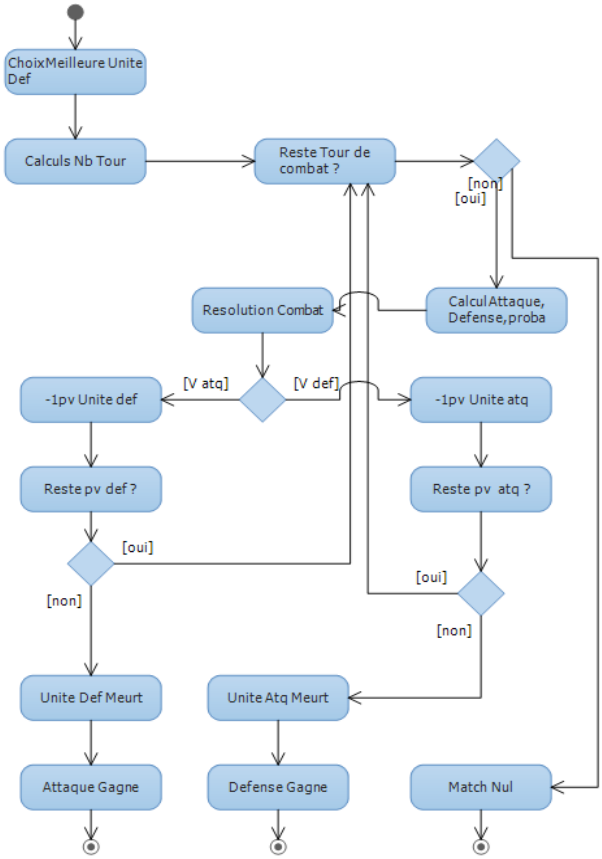

Illustration 3: Diagramme d'activité des combats

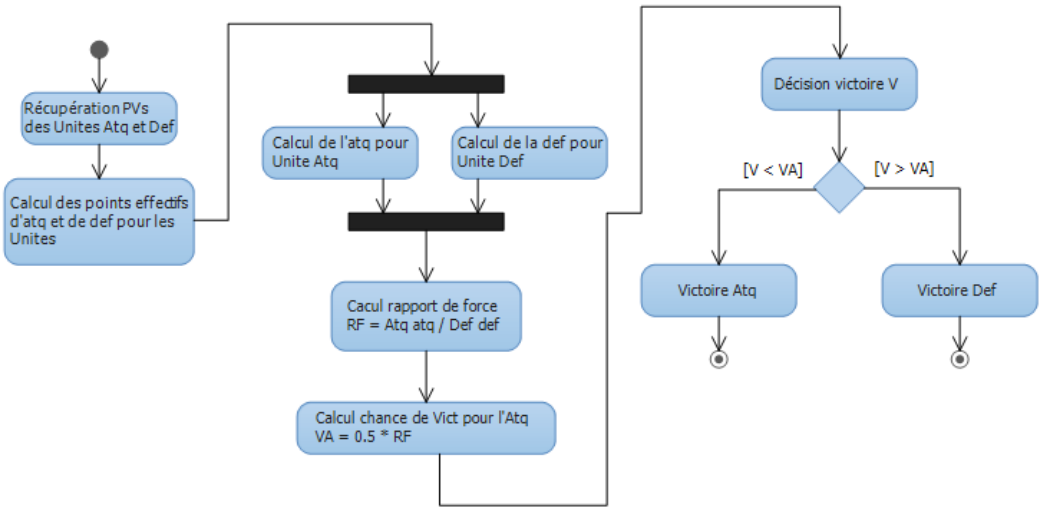

Illustration 4: Diagramme d'activité du calcul des probabilités d'un combat

Déroulement d'une partie

L'état d'une partie à un instant donné dépend de nombreux paramètres : quel est le joueur jouant actuellement ? Y-a-t-il un combat en cours ? Qui gagne ? Le diagramme d'état-transition ci-dessous (Illustration 5) permet une meilleure visualisation de cet état.

Au cours de son tour de jeu, un joueur doit utiliser toutes ses unités avant de pouvoir accumuler des points. Cette utilisation peut être soit un simple déplacement, soit générer un combat. Lorsqu'il a fini son tour, c'est à l'autre joueur de jouer, si toutefois il reste des tours de jeu.

Il y a deux possibilités de terminer une partie : soit un joueur n'a plus d'unité, auquel cas son adversaire à gagner ; soit le nombre maximum de tour a été joué, auquel cas le vainqueur est celui qui a gagné le plus de points au cours de la partie.

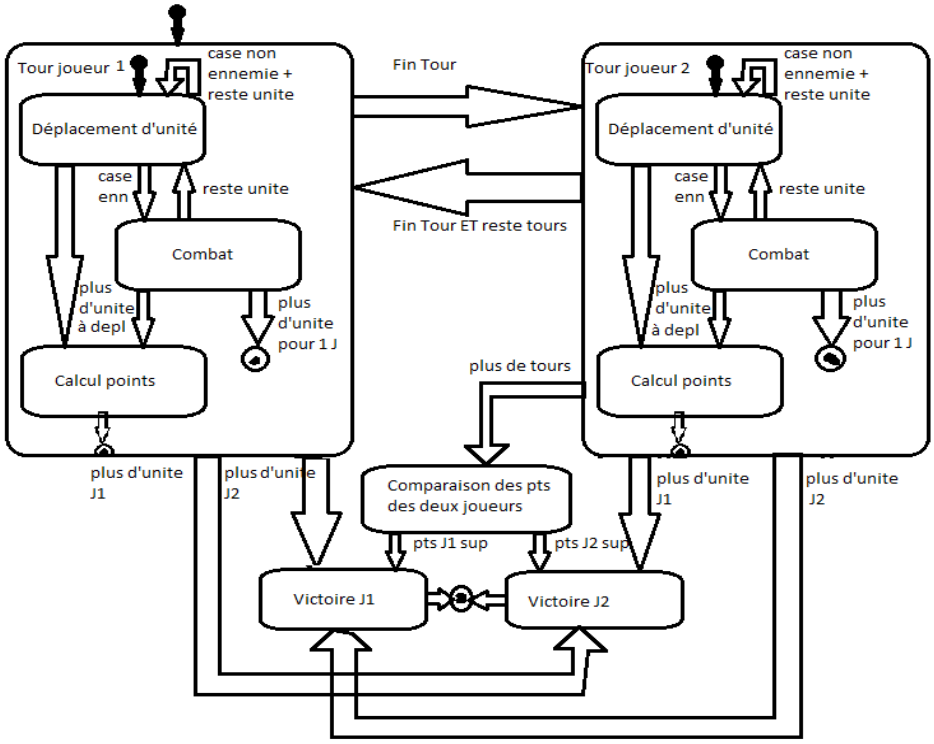

Illustration 5: Diagramme d'état-transition de la partie

Cycle de vie d'une unité

L'état d'une unité ne varie qu'en fonction de ses points de vie, son attaque et sa défense propre n'étant jamais modifiées. Le diagramme d'état-transition ci-dessous (Illustration 6) montre les différents états possible d'une unité. Un grand état « PV >= 1 » pourrait être fait au lieu des 5 états différents dans le super-état Combat, cependant les points d'attaque et de défense effectifs varient avec les points de vie, aussi faire apparaître chacun de ces états n'est pas dénué d'intérêt.

Le super-état Combat est historique afin de reprendre à chaque nouveau combat en se plaçant sur l'état représentant le nombre de points de vie restant de l'unité à la fin du précédent combat. De plus, la destruction de l'unité s'effectue dès que celle-ci se retrouve dans l'état « Mort », c'est-à-dire qu'elle ne possède plus de point de vie.

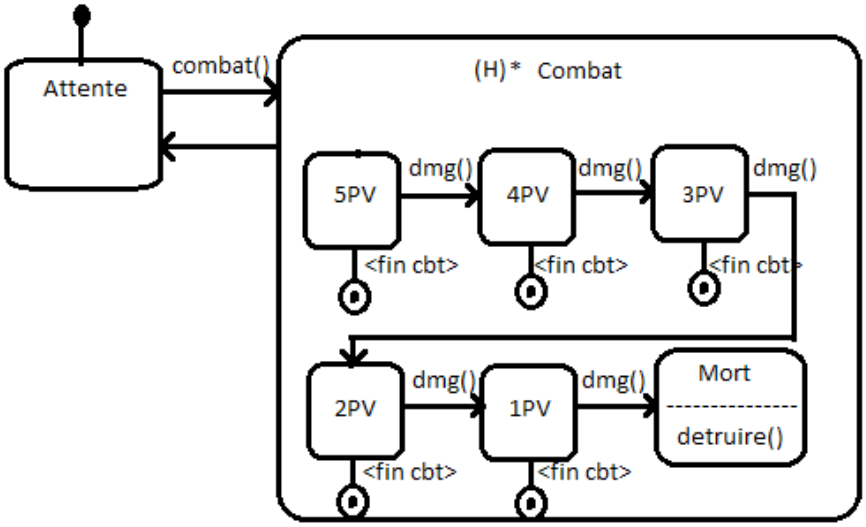

Illustration 6: Diagramme d'état-transition d'une unité

Diagramme de classe

Ce diagramme (Illustration 7) informe la manière dont le jeu sera implémenté en détails : on y retrouve les différentes classes, rangées de sorte à faire ressortir les patrons de conception que nous avons utilisées.

En haut à gauche se trouve le patron de Stratégie Plateau, qui nous servira à construire le plateau et à le remplir en fonction du type de partie choisie : en effet les algorithmes pour remplir une carte n'est pas le même selon sa taille afin d'optimiser le rapport coût-temps de création de celle-ci.

En haut à gauche se situe la partie concernant les cases dont sera composé le plateau. Afin d'optimiser l'utilisation mémoire à la fois à la création et lors de la partie, la patron de poids-mouche est utilisé : Chaque type de case ne sera créé qu'une seul fois par la fabrique de case, puis le plateau ne fera que savoir via une matrice quel le type d'une case. Ainsi chaque case ne sera pas très lourde.

Par ailleurs, puisque projet comporté originalement des bonus sur certaines cases, le choix avait été fait de réaliser également un poids-mouchage de ceux-ci, tout en les considérant que des décorations des cases. Cette implémentation sera réalisée si possible selon l'avancement du projet, mais nous avons décidé de garder la trace de notre réflexion à ce sujet.

En bas à droit se situe tout ce qui constitue le jeu en dehors des classes de modélisation du jeu « physique ». On y retrouve un singleton Combat qui sera responsable de la résolution des combats, ceux-ci se déroulant un par un et toujours de la même manière. Il contiendra entre autre

l'algorithme de résolution des combats vu précédemment.

Nous retrouvons également le patron de montage afin de créer une partie. Il n'est pas nécessaire de créer une classe pour chacun des types de partie : le seul changement se situant à la création du plateau, le monteur appellera donnera simplement la bonne stratégie au plateau.

La classe joueur est également présente afin de compter les points, connaître l'appartenance des unités etc.

En bas à droite se situe la fabrique d'unité selon le peuple choisi. Il n'est pas nécessaire de créer une classe fabrique : chaque peuple ne construisant qu'un seul type d'unité, et surtout ces unités n'étant créée qu'une seule fois au début de la partie, la méthode de créations de celles-ci peut être située directement dans la classe peuple.

La gestion de la vue n'est pas visualisable ici car il existe une méthode spécifique au C# qui n'est pas représentable sur un diagramme de classe.Diagrammes de séquences

Utilisation d'une unité

Ce diagramme (Illustration 8) montre la manière dont sera géré une unité au cours d'un tour. Plusieurs cas sont présentés ici, au travers des cadres de séquence : en effet, certaines actions dépendent de l'environnement, comme le fait de vouloir se déplacer vers une case occupée par un ennemi entraîne finalement un combat. Ainsi le message <<bool amie (?) >> est traité selon sa réponse dans les cadres suivants.

Création d'une partie

Ce diagramme () exprime la méthode de création d'une partie en mettant en avant les appels successifs aux fonctions.

Illustration 7: Diagramme de classe du projet SmallWorld

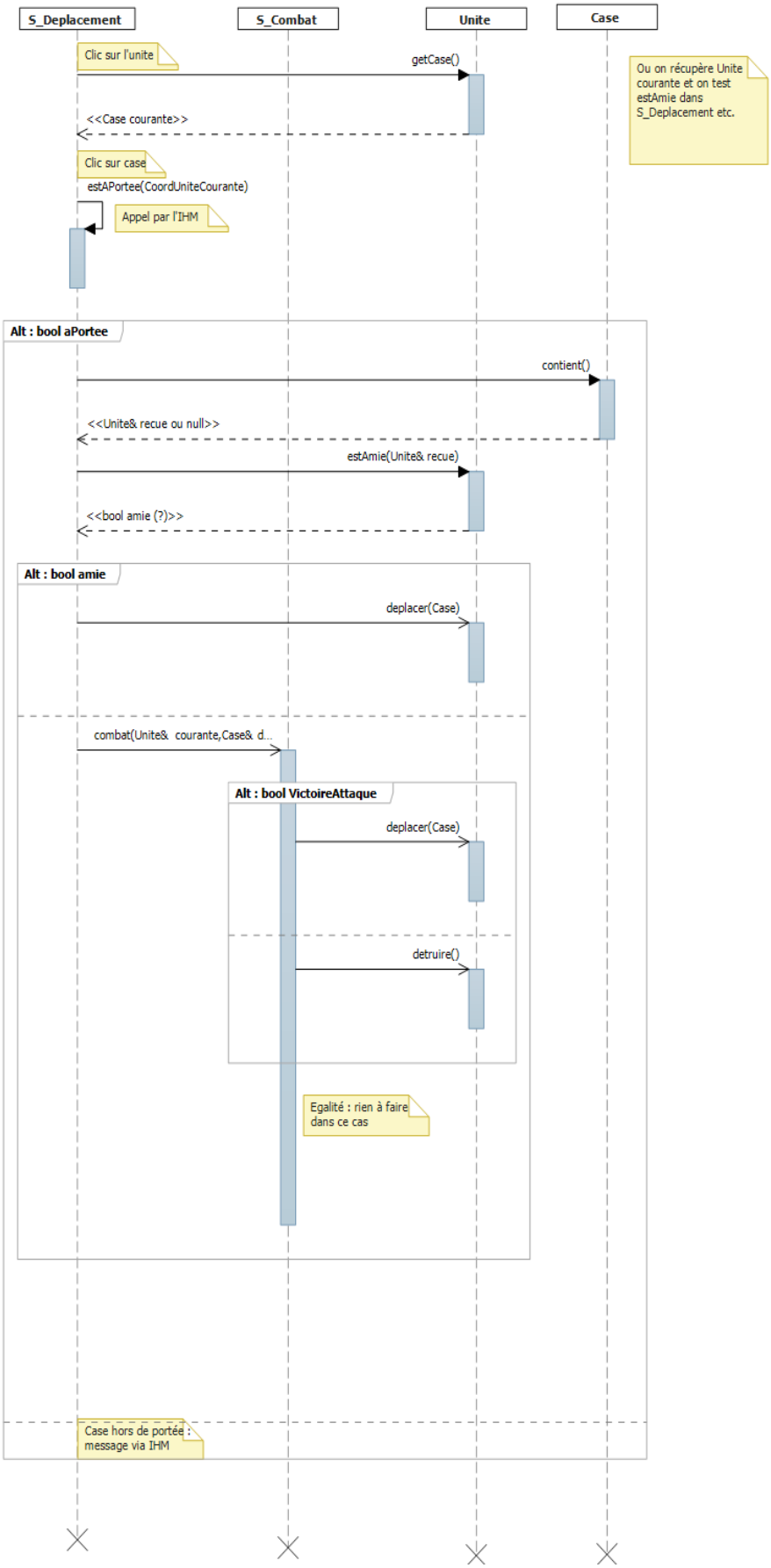

Illustration 8: Diagramme de séquence d'utilisation d'une unité