Web Game Engine - Cahier des charges

Ilyas Boutebal Maxime Colin Adrian Gaudebert
Youness Hamri Van-Duc Nguyen

2 décembre 2010

Table des matières

Chapitre 1

Contexte

1.1 Le Jeu vidéo

Le marché du jeu vidéo s'est beaucoup développé depuis l'arrivée de l'ancêtre Pong ou du célèbre Tetris. On joue aujourd'hui à des jeux de plus en plus beaux, demandant de plus en plus de ressources à nos machines. Mais on voit aussi d'autres utilisations du jeu vidéo arriver : le Serious Gaming, par exemple, est une branche en pleine expansion en ce moment. Le Social Gaming également, représenté activement par les nombreux jeux basés sur Facebook et ses possibilités en terme de diffusion.

Parmi les différentes "catégories" de jeux vidéos, il en est une qui s'est particulièrement développée ces dernières années : les jeux en ligne. Dans cette grande catégorie, on trouve beaucoup de types de jeux différents : les MMO dans le genre de World of Warcraft, les jeux multijoueur comme Counter Strike ou Team Fortress, les modes multi de jeux principalement solo, et les jeux par navigateur.

1.2 Les jeux par navigateur

Le jeu par navigateur, ou Browser Game en anglais, se joue par définition dans un navigateur Internet. On retrouve donc dans cette catégorie les jeux "Facebook" cités plus haut, la majorité des jeux en Flash, mais également un très grand nombre de jeux que nous appellerons les Jeux Web, puisqu'ils se basent sur les technologies du Web ouvert. Quelques exemples de jeux web relativement connus : Ogame, Travian, ou le français Hordes.

Ces trois exemples ne sont cependant pas représentatifs de la diversité que l'on trouve dans les jeux web. Les jeux d'élevage virtuel, par exemple, sont extrêmement nombreux sur le Web. À quoi est due cette profusion de jeux web ? Majoritairement à la simplicité d'accès du développement web. Faire un jeu web, en soit, c'est développer un site web. Or, avec des technologies très répandues comme PHP, HTML et CSS, avec toutes les ressources que l'on trouve autour de ces dernières (il suffit de regarder les tutoriels du Site du Zéro pour s'en rendre compte), il est très simple pour une personne un peu motivée de créer son propre jeu web.

Malheureusement, s'il est simple de créer un jeu web "basique", les développeurs sont rapidement limités par les technologies qu'ils utilisent. Il est quasiment impossible de faire du temps réel avec PHP, le couple HTML / CSS n'est pas adapté à l'affichage d'effets spéciaux ou d'animations complexes, et si l'arrivée récente de frameworks JavaScript comme jQuery a permis de repousser un peu ces limites techniques, cela ne résout pas le problème.

1.3 Les technologies du web

Il existe une solution simple aux limitations techniques actuelles des technologies web : se tourner vers le futur et utiliser de nouvelles technologies, pas encore éprouvées, mais qui permettent d'aller beaucoup plus loin dans la création de nos jeux web.

HTML 5, la nouvelle mouture du langage de description des pages web, apporte au développeur un très grand nombre de fonctionnalités clés : le temps réel dans le navigateur avec les WebSockets, les manipulations graphiques avancées avec le SVG et les Canvas, la vidéo et le son avec Video et Audio, et bien d'autres.

La version 3 du langage CSS offre elle aussi d'alléchantes nouveautés : les animations, les transitions, les polices particulières, ainsi que tous les effets graphiques amplement simplifiés.

Toutes les avancées actuelles du web ouvrent indéniablement les portes à des jeux de plus en plus impressionnants, de plus en plus profonds, le tout se passant dans un navigateur ! Il faudra cependant du temps pour que les développeurs maîtrisent tout ceci, pour que des outils facilitant l'utilisation de ces technologies apparaissent, et donc pour que l'accès à toutes ces possibilités pour nos jeux devienne simple.

1.4 Besoin

Ce besoin est bien réel : le projet Fightly[1], par exemple, vise à créer un jeu de stratégie au tour par tour bénéficiant directement de ces avancées. La création d'un outil fournissant un ensemble de fonctionnalités de base, facilitant l'accès aux technologies Web récentes et simplifiant donc largement le développement d'un tel projet ne pourra que mener à une amélioration globale de ce jeu.

Fightly est un jeu de stratégie au tour par tour dans lequel des joueurs s'affrontent en manipulant des unités. Les parties se déroulent sur des cartes à cases hexagonales, les unités s'y déplaçant en fonction du terrain et des stratégies des joueurs. Les joueurs choisissent au début de la partie leurs unités, composées de fantassins, archers ou chevaliers, chacunes ayant des caractéristiques et des capacités différentes. Le jeu se déroule en temps et en tours limités, le gagnant étant le dernier joueur en vie, ou à défaut celui ayant le plus de points.

Il existe également le jeu Alonya[2], qui reprend largement le concept des jeux Civilization en l'adaptant au Web (et en ajoutant quelques modifications bien sur). Ce jeu est actuellement en développement, et n'utilisera donc a priori pas

[1] http://fightly-dev.lqbs.fr/forum/
[2] http://alonya.eu/

notre moteur, mais l'existance d'un tel outil aurait certainement aidé la création d'Alonya.

Chapitre 2

Le projet

2.1 Objectifs

Voici donc où se place ce projet : notre objectif est de fournir un outil complet, permettant de créer de façon simple et rapide un jeu profitant des dernières avancées en matière de technologie web. Il constituera un ensemble d'outils qui fournissent un cadre de développement et des fonctionnalités permettant d'implémenter tout ou partie d'un jeu.

Le moteur sera prioritairement conçu pour des jeux de stratégie au tour par tour. On cherchera à le rendre aussi générique que possible, afin de permettre une grande diversité dans les types de jeux développables avec notre outil. On veillera aussi à faciliter l'extensibilité du projet, toujours dans le but d'offrir la possibilité d'en étendre facilement les fonctionnalités.

Un des objectifs secondaires du projet est d'utiliser des technologies récentes et innovantes. Si la compatibilité avec les différents navigateurs du marché doit être un facteur important, elle ne doit pas être une limitation dans le choix des outils à utiliser.

Pour finir, ce projet sera libre, c'est-à-dire qu'il sera distribué sous licence libre, et qu'à terme on voudra mettre en place une communauté autour du développement de ce produit.

2.2 Les jeux de stratégie au tour par tour

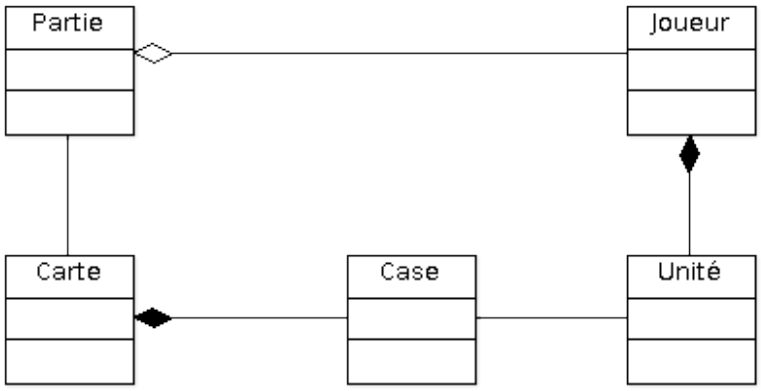
L'objectif de ce projet est de permettre la création de jeux de stratégie au tour par tour : on va donc étudier ce type de jeux pour en dégager les grands concepts, les particularités, les éléments nécessaires. Nos choix se sont portés sur les jeux suivants :

- Dofus : MMORPG dont le module de combat est en tour par tour
- Civilization : Jeu de gestion d'un empire
- Heroes of Might and Magic : Jeu de campagnes militaires, univers médiéval / fantastique
- Battle for Wesnoth : Jeu d'aventure, univers médiéval / fantastique
- Fightly

Les diagrammes représentant les concepts de ces jeux, ains que des informations complémentaires, sont disponibles dans l'Annexe A. De ces jeux, nous

avons donc trouvé un ensemble de concepts qui reviennent, et composent donc a priori la base de tout jeu de stratégie au tour par tour. Certains de ces éléments n'ont pas forcément le même nom selon le jeu, et des différences au niveau de l'implémentation peuvent apparaitre, mais on peut les généraliser en un concept plus global.

Voici donc le diagramme modélisant les concepts qui nous paraissent essentiels à la réalisation d'un jeu de stratégie au tour par tour :



2.3 Spécifications

L'objectif de ce projet est de créer un moteur de jeu web : on utilisera donc naturellement une architecture Client / Serveur.

La partie Serveur devra être mise en place sur un serveur dédié. Elle prendra en charge les règles du jeu, les parties et leur état (appelé le "Monde"), les joueurs. Elle gérera également la communication avec les clients. Cette partie devra donc, la majorité du temps, recevoir des demandes d'action de la part d'un joueur, valider cette action par rapport aux règles et à l'état du Monde de la partie concernée, puis valider cette action et répercuter les changements induits chez tous les joueurs. On devra également gérer la mise en place d'une partie (par exemple, dans le jeu Fightly, il faut attendre d'autres joueurs, puis chaque joueur créé une armée, puis chaque joueur place son armée sur la carte, et enfin le jeu commence "réellement") et la fin d'une partie (calcul du gagnant, actions particulières en fonction du jeu).

Cependant, on ne s'occupera pas dans cette partie Serveur de tout ce qui correspondrait au rôle d'un serveur web classique. Par exemple, on n'assurera pas la distribution des fichiers statiques (c'est-à-dire HTML, CSS, JavaScript et images), ni la création et configuration des parties. On considère que, lorsqu'un joueur rejoint une partie, il a déjà tous les éléments nécessaires à sa gestion. Ceci implique donc qu'un deuxième serveur, gérant la distribution des fichiers statiques, soit mis en place : mais ce n'est pas une contrainte forte, puisqu'un jeu Web contiendra forcément un serveur Web, au moins pour une aide de jeu ou un forum. Notre moteur ne sera donc pas indépendant, mais s'intégrera à une application Web existante, sur laquelle il se basera pour fonctionner.

La partie Client correspond à ce qui est exécuté dans le navigateur du joueur. Cette application gérera donc l'affichage du jeu chez le joueur, ainsi que les interactions avec ce dernier. L'application cliente devra pouvoir communiquer avec le serveur, mais également avec d'autres clients (typiquement, les autres joueurs présents dans une partie en cours). En règle générale, cette partie enverra des demandes d'action au serveur, et appliquera en local les changements qui lui sont ordonnés par ce même serveur.

Chapitre 3

Conception

3.1 Architecture générale

Diagrammes UML
Diagramme de classe du Serveur :

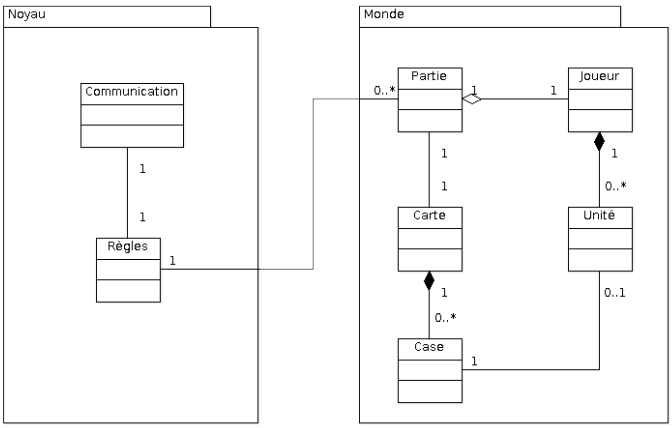
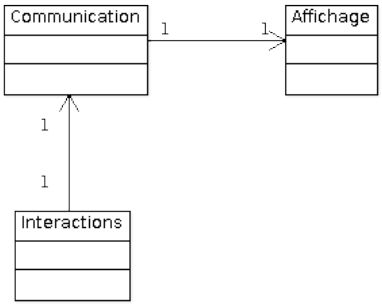


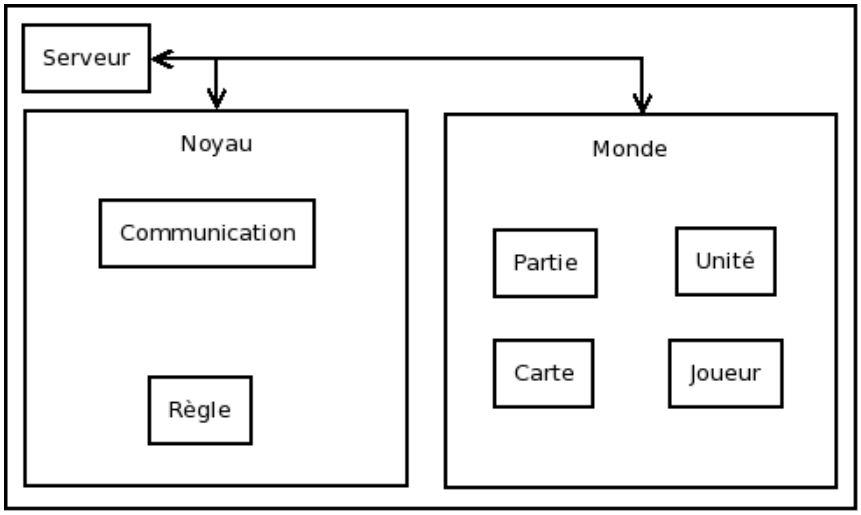
Diagramme de classe du Client :



3.2 Serveur

Le serveur sera la partie du moteur qui fera tourner les parties et permettra la communication et les interactions entre les joueurs. Il aura pour rôle de contrôler les parties, lier les clients et gérer leur actions. Il aura également le rôle d'arbitre dans une partie.

Le serveur sera séparé en deux grandes parties : une partie Noyau et une partie Monde.



3.2.1 Package Noyau

La partie Noyau du serveur gère les fonctionnalités de base du moteur. Elle contient un ensemble de modules qui sont a priori indépendants du type de jeu implémenté. L'objectif principal de ce package est de faire le lien entre les actions du joueur et la représentation centralisée du Monde[1] d'une partie. On trouve donc deux parties principales, une gérant les liens avec les clients, l'autre s'occupant des règles et du lien avec la gestion du Monde.

Le module communication du serveur prend en charge tous les échanges réseau avec les clients. Elle s'occupe d'accepter les nouvelles connexions, fait l'association entre client (au sens réseau) et joueur (au sens du jeu), et permet donc de recevoir des messages d'un joueur ainsi que de lui en envoyer. Ce module aura aussi à charge la gestion des déconnexions éventuelles de joueurs.

Le module règles sera chargé de stocker et de traiter les règles du jeu. Il contiendra des fonctionnalités permettant de créer une base de règles. Ces règles pourront être ajoutées ou supprimées individuellement ou chargées à partir d'un fichier de règles. La partie règle devra également valider (ou invalider) les actions demandées par les joueurs, en fonction des règles connues et de l'état du Monde.

3.2.2 Package Monde

La partie Monde est composée d'un ensemble de modules qui gèrent les fonctionnalités principales d'un jeu de stratégie au tour par tour. On y retrouvera donc les notions que l'on a dégagées des différents jeux étudiés (cf. partie 2.2 Les jeux de stratégie au tour par tour), et que l'on considère comme étant primordiales pour tout jeu STpT[2].

Ce package devra être le plus générique et le plus extensible. On devra pouvoir modifier facilement le comportement des modules sans que cela n'affecte les autres. De plus, on devra pouvoir ajouter de nouvelles fonctionnalités, et les intégrer au moteur.

Dans ce package Monde, on trouvera quatre modules principaux. Tout d'abord, un module représentera les parties, en permettant de les charger depuis un support de persistence. Il gérera également les tours, l'ordre des joueurs, et toute la configuration particulière qu'une partie peut avoir.

Le deuxième module concernera les joueurs : on les caractérisera pour les différencier, et chaque joueur possèdera un ensemble d'unités qui pourra manipuler. On stockera également les points gagnés par le joueur durant la partie.

Un troisième module sera dédié à la gestion de la carte. Celle-ci sera composée d'un ensemble de cases, chacune de ces cases étant caractérisée par plusieurs paramètres, comme l'accessibilité ou l'influence sur les unités qui s'y trouvent.

Enfin, on aura un module pour représenter les unités. Une unité possèdera un ensemble de paramètres (par exemple, une valeur offensive, une valeur défensive, une vie...), appartiendra à un unique joueur et sera à tout moment sur une seule case. Une unité détruite sera supprimée de la partie, et donc retirée de l'état du Monde.

On choisira d'implémenter ces différents modules de telle sorte qu'ils correspondent globalement aux besoins du jeu Fightly présenté précédemment. Ainsi, on aura des modules relativement générique, Fightly étant un jeu qui exploite

[1] Rappel : on appelle "le Monde" l'ensemble des données représentant l'état d'une partie.
[2] STpT : Stratégie au Tour par Tour

des mécaniques de jeu très répandues. On fera cependant attention à être le moins spécifique possible, ou à faire en sorte que les parties spécifiques soient isolées et donc facilement interchangeables, afin de simplifier le passage à un autre type de jeu.

3.3 Client

Le client doit gérer l'affichage (et éventuellement la gestion du son), la réception de données envoyées par le serveur, et les données entrées par le joueur (inputs) et l'envoi des demandes de modification des données (modification du monde).

3.3.1 Communication

Le client doit être capable d'établir une connexion avec le serveur après l'identification du joueur, il peut également établir des connexions avec les autres clients pour échanger des messages (le Chat par exemple).

3.3.2 Affichage

Pour l'affichage, La partie client de notre moteur de jeux ne dispose que les données d'une partie de la carte, c'est la zone ou le périmètre des unités du joueur. A chaque déplacement d'unité, le client doit demander au serveur de charger la partie correspondante de la carte.

La carte est représentée sous forme de cellules (rectangulaire ou hexagonale) de plusieurs types (montagne, foret, chemin, mer ...), chaque cellule a des propriétés (accessibilité : oui ou non par une unité par exemple).

Deuxième grand élément du jeu c'est les unités, le client dispose d'un ensemble de fonctionnalités capable de : Créer, déplacer et mettre a jour les propriétés de ces unité ainsi que la suppression d'une unité (après une éventuelle déconnexion).

3.3.3 Input

Cette partie s'occupe de la lecture des périphériques externes comme :
- La Souris (lecture de l'état des boutons (clics) et de la position).
- Le Clavier (de l'état des boutons).

Ces Inputs seront utilisés directement par les fonctionnalités de

Chapitre 4

Livrables intermédiaires

On fournira au client trois livrables intermédiaires au cours du projet

4.1 Veille technologique

On effectuera, dans une partie mesure, une veille technologique afin d'étudier l'existant et de déterminer quels sont les meilleurs choix de technologies à utiliser pour ce projet.

Date de rendu : 17 décembre 2010.

4.2 Dossier d'analyse

Il sera important d'écrire un dossier de conception détaillé, présentant tous les aspects techniques du projet, les problématiques et les solutions envisagées. Ce dossier servira de base au développement du projet, et devra être validé par le client.

Date de rendu : 7 janvier 2011.

4.3 Version de démonstration

On voudra, à la moitié de la phase de développement, fournir au client une version de démonstration au client. Cette application consistera en un simple client "spectateur", qui ne fera donc rien d'autre qu'afficher le Monde d'une partie. Le client pourra donc valider ou imputer nos choix à ce moment, et nous permettre de corriger certaines erreurs éventuelles avant la fin du projet.

Date de rendu : 19 janvier 2011.

Chapitre 5

Livrables finaux

À la fin du projet, on fournira au client les trois éléments suivants :

- Le code source complet de l'application développée ;
- Une documentation technique, à destination des développeurs qui utiliseront le moteur, mais également à l'intention des développeurs qui contribueront par la suite au projet, celui-ci étant libre ;
- Une démonstration technique du moteur.

Date de rendu : 4 février 2011.

Chapitre 6

Organisation

6.1 Décomposition générale

Décomposition du projet en modules

- Serveur
    - Noyau
        - Communication
        - Règles
    - Monde
        - Partie
        - Joueur
        - Carte
        - Unité
- Client
    - Affichage
    - Communication
    - Interactions

6.2 Responsabilités

La conduite du projet sera assurée par Adrian. Il aura par ailleurs à développer le module Affichage de la partie Client.

Maxime sera responsable de la partie Serveur, il devra donc en contrôler le bon avancement, la cohérence et la qualité au fur et à mesure du développement. Il sera également en charge du développement du module de Règles.

Le package Noyau du serveur sera sous la responsabilité de Duc, qui devra donc en assurer le suivi, et qui développera le module Communication de ce même noyau, ainsi que celui de la partie Client.

Le package Monde sera pris en charge par Youness. Il devra également développer les modules Partie, Joueur et Carte du Serveur.

Enfin, la partie Client sera sous la responsabilité de Ilyas, qui en développera également le module Interactions. Il développera aussi le module Unité du package Monde du serveur.

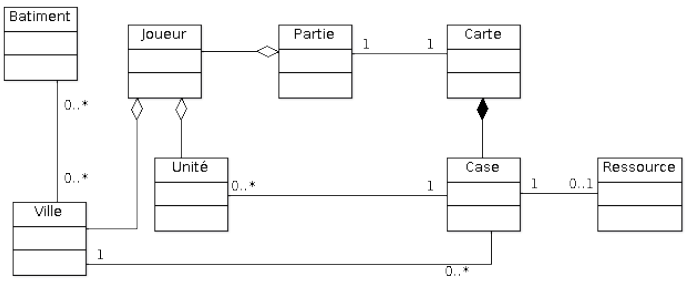
6.3 Planning

Planning

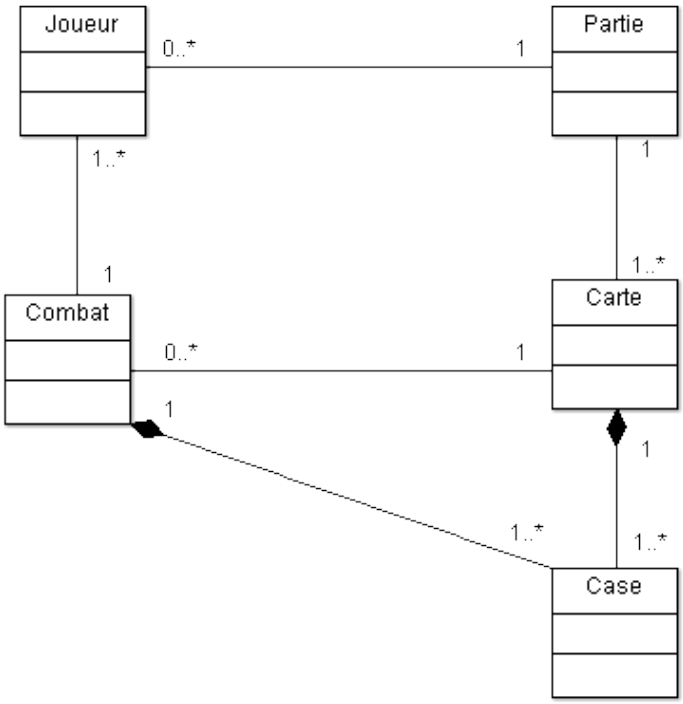
Annexe A

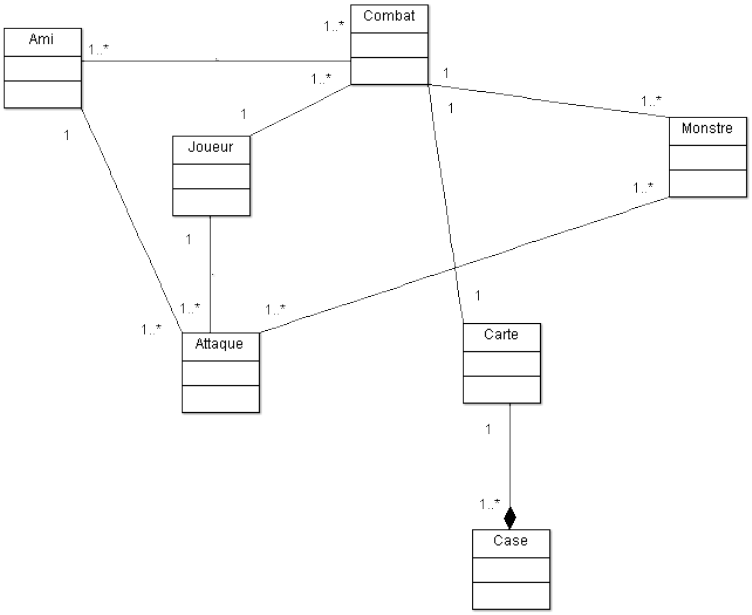
Jeux de stratégie au tour par tour

A.1 Civilization



A.2 Dofus


Joueur
Partie
0..*
1
1..*
1
1
1..*
Combat
Carte
0..*
1
1
1
1..*
1..*
Case


Ami
Combat
1..*
1..*
1
1
1..*
1
1
Joueur
Monstre
1..*
1
1..*
1..*
1..*
Attaque
Carte
1
1..*
Case

A.3 Heroes of Might and Magic

Diagramme de classes présentant les principaux concepts du jeu :

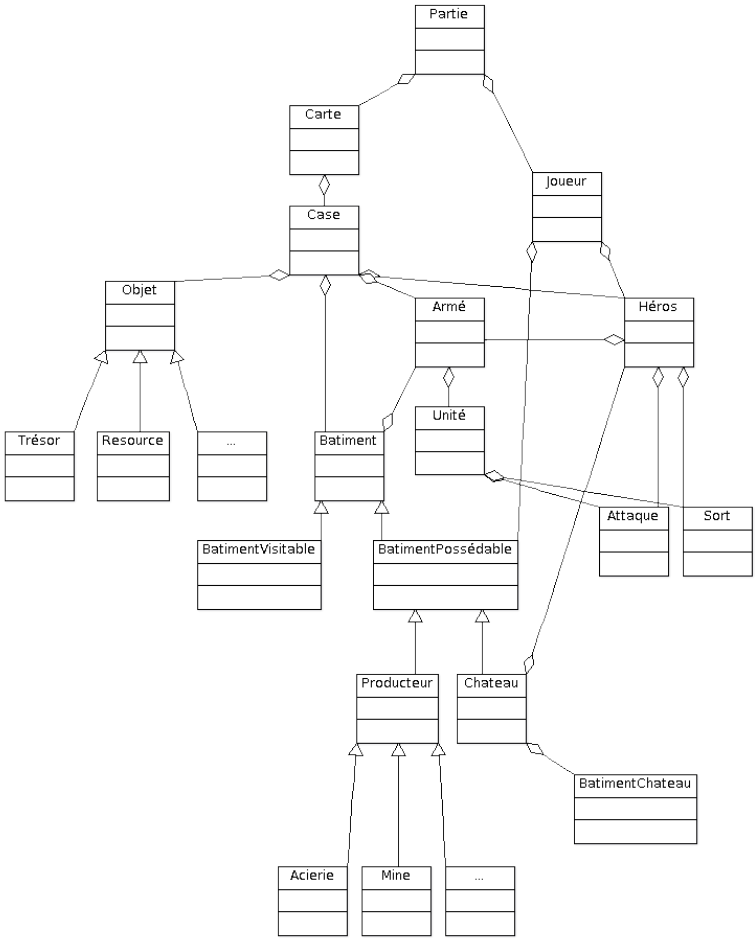
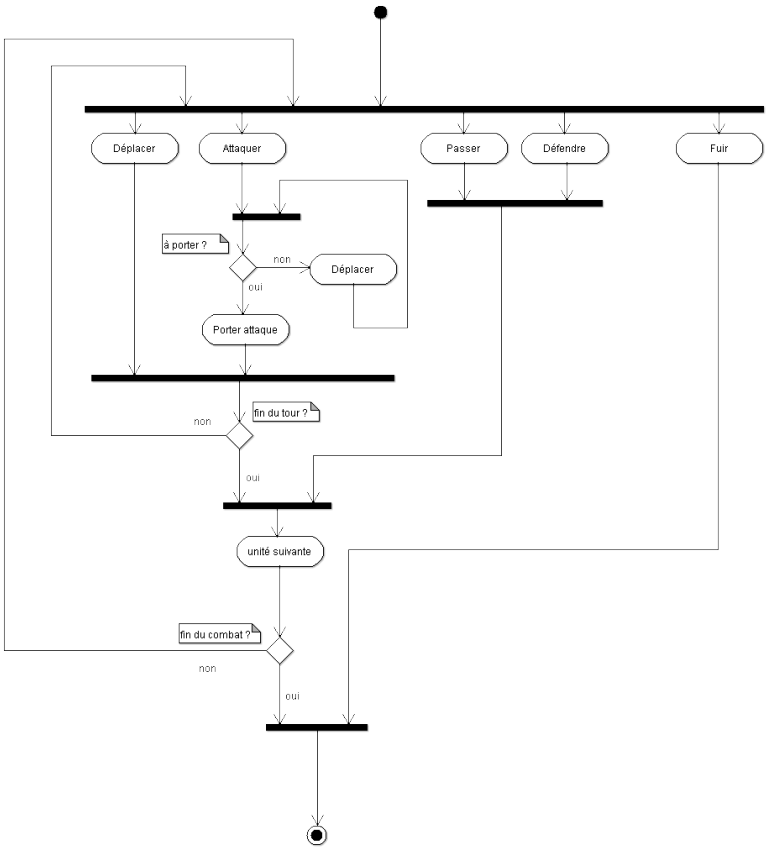
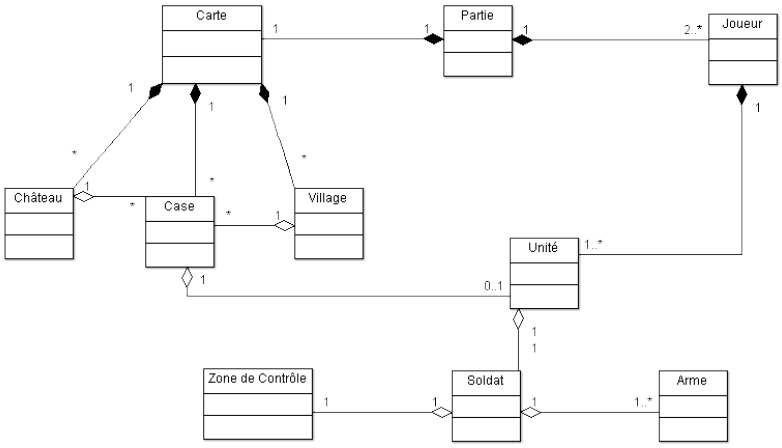


Diagramme d'activités présentant le déroulement global du jeu :



A.4 Battle for Wesnoth



A.5 Fightly