Projet d'UML : Ring

(IUT, département informatique, 1re année)

Laurent AUDIBERT

L'objectif du projet consiste à proposer un modèle UML d'une petite application permettant de mettre en œuvre des combats entres deux personnages, puis de proposer une implémentation en Java de cette application.

1 Description du jeu

1.1 Description des combattants

Les combattants peuvent être de trois types : **guerrier**, **athlète** ou **magicien**. Tous les combattants du jeu sont caractérisés par un nom, une force physique, une dextérité, une intelligence, une capacité de concentration, une vitalité et une expérience.

La vitalité d'un personnage est son nombre de points de vie. Ce nombre décroit au fur et à mesure qu'un personnage est blessé au combat. Le personnage meurt quand sa vitalité devient nulle.

L'expérience d'un combattant augmente quand il gagne un combat et décroit quand il perd, avec une valeur plancher et une valeur plafond.

Le tableau de la figure 1 résume les aptitudes des trois types de combattants qui doivent être mises-en-œuvre dans le jeu.

Chaque personnage possède un certain nombre de capacités (armes, sortilèges...) décrites dans la section suivante (1.2).

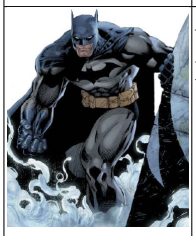
Caractéristiques communes	Dans tous les cas, les équilibres suivants doivent être respectés : • $force + dextérité + intelligence + concentration \leqslant 100 + expérience$ • $1 \leqslant expérience \leqslant 20$ Au début d'un combat, la vitalité est initialisée de la manière suivante : • $vitalité = 200 - (force + dextérité + intelligence + concentration) + expérience \times 3$
	Un **guerrier** est un combattant qui brille plus par sa force physique que par son intelligence. La contrainte suivante est imposée à tous les guerriers : • $force \geqslant dextérité + 10 \geqslant intelligence + 10 \geqslant concentration$
	Un **athlète** est un combattant assez complet capable aussi bien de manier l'épée que la magie. Les contraintes suivantes sont imposées à tous les athlètes : • $force \geqslant 20$ • $dextérité \geqslant 20$ • $intelligence \geqslant 20$ • $concentration \geqslant 20$
	Bien entendu, la force d'un **magicien** réside dans sa capacité à faire de la magie, au détriment de sa force physique. Les contraintes suivantes sont imposées à tous les magiciens : • $intelligence \geqslant max(force, dextérité) + 15$ • $concentration \geqslant max(force, dextérité) + 15$

FIGURE 1 – Description des aptitudes des trois types de combattants

1.2 Description des capacités

Les capacités permettent aux combattants de porter des coups, de parer des coups et de se soigner. Elles se subdivisent en trois catégories : **attaque**, **parade**, **soin**. Le nombre de capacités d'un combattant doit être inférieur à son niveau d'expérience divisé par deux, mais ne peut descendre en dessous de deux.

Toutes les capacités fonctionnent selon le même principe :

1. Vérifier sa réussite – La probabilité de réussite de la mise-en-œvre d'une capacité est fonction de l'aptitude du combattant ainsi que des caractéristiques de la capacité.
2. Mesurer l'efficacité – Si la mise-en-œvre d'une capacité a réussi, son efficacité est fonction de l'aptitude du combattant ainsi que des caractéristiques de la capacité, sinon son efficacité est nulle bien entendu.

Les capacités possèdent un certain nombre de caractéristiques de type entier. La somme de ces caractéristiques doit toujours être égale à 100 et aucune caractéristiques ne peut être inférieure à 20. Le tableau de la figure 2 détaille toutes les catégories de capacités des combattants.

1.3 Naissance d'un nouveau combattant

Pour créer un nouveau combattant, il faut :

1. Choisir un type de personnage (guerrier, athlète ou magicien) et son nom
2. Choisir la valeur de ses aptitudes (force, dextérité, intelligence, concentration) en respectant les contraintes détaillées dans le tableau de la figure 1
3. Lui affecter une expérience de 1
4. Créer 2 capacités en respectant les contraintes exposées dans la section 1.2, puis les lui affecter

Nom	Catégorie	Caractéristiques	Probabilité de réussite	Impact	Description
Épée	Attaque, Parade	impact parade maniabilité	Attaque : $dextérité \times maniabilité/10000$ Parade : $dextérité \times maniabilité/10000$	Attaque : $force \times impact/100$ Parade : $force \times parade/100$	À la fois arme offensive et défensive son efficacité est divisée par 3 en cas de parade d'un sortilège
Sortilège offensif	Attaque	facilité efficacité	$concentration \times facilité/10000$	$intelligence \times efficacité/100$	C'est l'arme des magiciens
Bouclier	Parade	maniabilité protection	$dextérité \times maniabilité/10000$	$force \times protection/100$	Son efficacité est divisée par 3 si l'attaque est un sortilège
Sortilège défensif	Parade	facilité efficacité	$concentration \times facilité/10000$	$intelligence \times efficacité/100$	Bouclier des attaques magiques, son efficacité est divisée par 3 si l'attaque ne l'est pas
Remède	Soin	facilité efficacité	$dextérité \times facilité/10000$	$dextérité \times efficacité/100$	Attention, c'est de l'automédication
Sortilège guérisseur	Soin	facilité efficacité	$concentration \times facilité/10000$	$intelligence \times efficacité/100$	C'est le remède des magiciens

FIGURE 2 – Description des catégories de capacités des combattants

1.4 Déroulement d'un duel

Un duel se déroule par tour de jeu. Un tour de jeu ne concerne qu'un seul combattant. De manière générale, un tour de jeu permet de faire 2 actions. Une action est la mise-en-œuvre d'une capacité, ou une capitulation. Une capitulation met fin à la partie sans qu'aucun combattant ne décède. Un exemple de tour de jeu classique comprend une parade et une attaque. Il est raisonnable de placer deux attaques si l'adversaire n'a pas réussi à en concrétiser une. Un duel se déroule de la manière qui suit.

Initialisation du duel

1. Calculer et affecter la vitalité à chacun des participants selon la formule présentée dans le tableau de la figure 1
2. Déterminer qui commence : c'est le plus expérimenté, ou le gagnant d'un tirage au sort en cas de niveau d'expérience identique.

Premier tour de jeux

3. Le combattant qui commence met en œuvre une capacité de son choix (généralement une attaque), la réussite et l'efficacité de cette capacité sont évaluées, affichées et appliquées dans le cas d'un soin
4. L'efficacité d'une attaque réussie est mémorisée pour impacter l'adversaire lors de son tour de jeu

Tour de jeu classique

5. Le combattant met en œuvre une capacité de son choix (par exemple une parade), la réussite et l'efficacité de cette capacité sont évaluées, affichées et appliquées dans le cas d'un soin ou d'une parade ; le combattant peut également capituler
6. Même déroulement que l'étape précédente (étape 5)
7. Le combattant perd autant de points de vie qu'il n'a pu parer, si sa vitalité devient nulle, le combattant décède et perd la partie
8. Si la partie n'est pas terminée, la somme de l'efficacité des attaques du combattant est mémorisée pour impacter l'adversaire lors de son tour de jeu
9. C'est au tour de jeu de l'adversaire en reprenant le déroulement à l'étape 5

L'application d'un soin consiste à incrémenter la vitalité du nombre de points d'efficacité d'un soin réussi.

1.5 Fin de partie

Une partie se termine quand l'un des deux adversaires décède ou capitule. Un combattant décédé l'est définitivement et ne pourra plus combattre.

Le vaincu perd un point d'expérience (sauf s'il n'en avait déjà plus qu'un) et éventuellement un point dans une aptitude tirée au sort de manière à respecter les contraintes du tableau de la figure 1. Le perdant peut également être amené à perdre une capacité de son choix pour que le nombre de ses capacités reste inférieur au nombre de ses points d'expérience divisé par deux (cf. section 1.2).

Les vaincueur gagne un point d'expérience (sauf s'il n'en avait déjà vingt) et éventuellement un point dans une aptitude de son choix tout en respectant les contraintes du tableau de la figure 1. Il peut également copier une capacité de son adversaire et l'ajouter aux siennes, ou remplacer l'une des siennes s'il en possédait déjà le maximum autorisé.

1.6 Fonctionnalités de l'application

L'application doit permettre de créer de nouveaux personnages comme décrit dans la section 1.3.

L'application doit également permettre de gérer des duels entre deux adversaires qui sont chargés depuis un fichier.

La sauvegarde des combattants est automatiquement gérée par l'application.

Si vous vous en sentez capable, vous pouvez créer une application permettant de jouer à deux avec deux ordinateurs distincts (l'application étant exécutée sur les deux machines) en utilisant un fichier d'échange (dont l'emplacement est paramétrable) pour synchroniser le déroulement d'un duel.

2 Partie UML du projet (Responsable : Laurent Audibert)

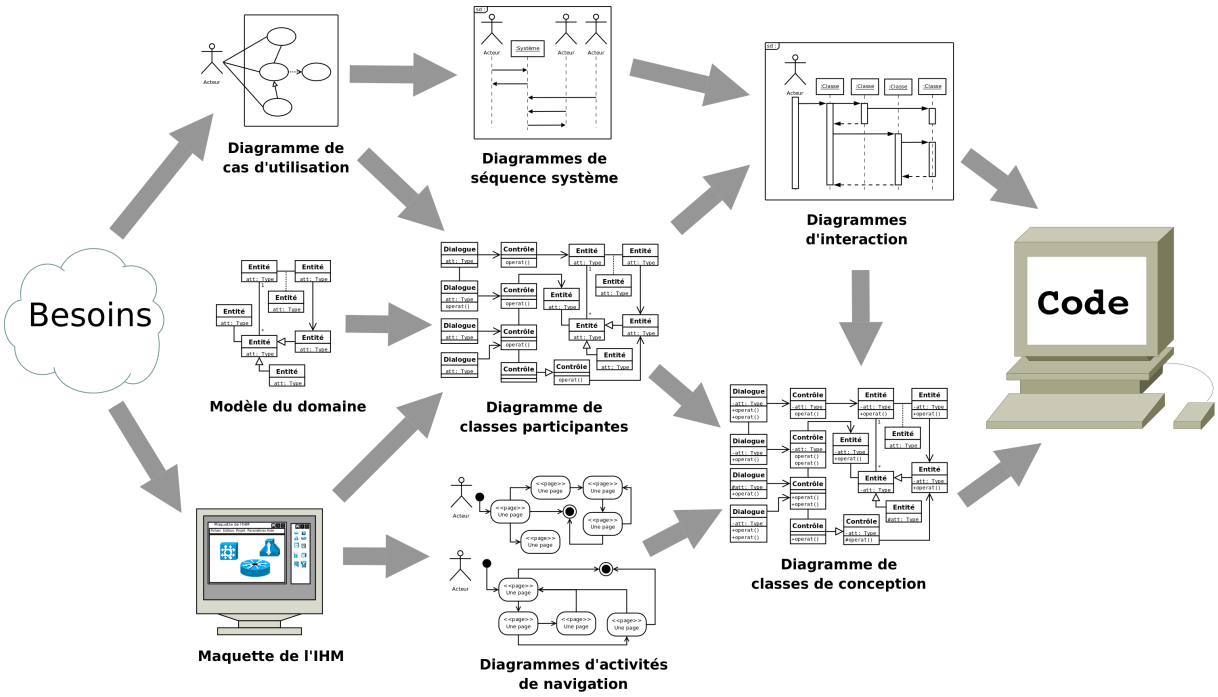


FIGURE 3 – Chaîne complète de la démarche de modélisation du besoin jusqu'au code.

2.1 Méthode de conception

Pour modéliser ce système, vous devrez utiliser la méthode minimale décrite en cours et synthétisée par la figure 3.

Dans le cadre du projet d'UML, vous n'irez pas jusqu'à l'écriture du code : l'implémentation concerne le projet de java. Toutes les autres étapes devront être prises en compte. Une manière de procéder pourrait donc être la suivante.

1. Identifiez et spécifiez les besoins en réalisant un *diagramme de cas d'utilisation* accompagné de la description textuelle des cas d'utilisation.
2. Illustrez les cas d'utilisation à l'aide de *diagrammes de séquence système*.
3. Analysez et modélisez le domaine en proposant un *modèle du domaine* sous forme de diagramme de classes.
4. Proposez une *maquette d'interface* graphique ou textuelle de l'application.
5. A l'aide d'un *diagramme d'activité de navigation* montrez la navigation dans l'interface graphique ou textuelle de l'application.
6. La première version du diagramme de classes (le modèle du domaine) doit être reprise pour y adjoindre les classes de contrôle et d'interface. Vous obtiendrez ainsi le *diagramme de classes participantes*
7. Chaque diagramme de séquence système doit être détaillé en prenant en compte les classes figurant dans le diagramme de classes participantes. Ces *diagrammes d'interaction* éclatent le système, qui n'est plus considéré comme une boîte noire, en plusieurs classes participantes de manière à montrer comment elles interagissent pour réaliser chacun des cas d'utilisation.
8. Les messages échangés entre les classes participantes dans les diagrammes d'interaction permettent de définir les opérations de chacune des classes participantes. Ces opérations seront spécifiées dans la nouvelle version du diagramme de classes : le *diagramme de classes de conception*. On veillera à ce que toutes les opérations soient réalisables, et que tout ce qui a été défini dans les activités de navigation soit présent.

Cette méthode est minimale et flexible. Si vous trouvez pertinent de spécifier certains aspects du logiciel en utilisant d'autres diagrammes, vous êtes naturellement libres de le faire.

Conseil : Pour ce projet, vous pouvez utiliser le patron de conception *Stratégie* très bien décrit dans l'article *Les limites de l'héritage : le pattern strategy* sur *le Site du Zéro* (`http://www.siteduzero.com/tutoriel-3-65563-les-limites-de-l-heritage-le-pattern-strategy.html`).

2.2 Modalités de remise du projet

Ce projet est à réaliser en binômes. Le rendu se fera impérativement sous forme de dossier papier à remettre le lundi 16 mai 2011 au secrétariat à l'attention de votre chargé de TD. Le 18 ou le 19 mai, vous recevrez par mail une confirmation de la bonne réception du projet. Si ce n'est pas le cas, inquiétez-vous !

3 Partie Java du projet (Responsable : Dominique Bouthinon)

3.1 Implémentation

L'implémentation java doit se faire obligatoirement après la modélisation UML et doit strictement respecter cette dernière. S'il s'avère que le modèle n'est pas adéquate, vous le modifierez avant de poursuivre l'implémentation. Toute rétro-analyse, consistant à concevoir le modèle UML à partir d'un programme java, sera très fortement pénalisée.

L'utilisation du polymophisme est obligatoire lorsqu'elle est possible. La clarté et la concision du programme seront considérées : le programme doit être commenté, les noms des classes, variables et méthodes seront évocateurs, enfin les méthodes contiendront peu de lignes.

L'affichage des menus doit être indépendant de la gestion du jeu.

3.2 Modalités de remise du projet

Le projet java sera remis le jeudi 9 juin selon des modalités fournies ultérieurement. Une soutenance individuelle aura ensuite lieu où le modèle UML, l'implémentation java et le travail de l'étudiant au sein du binôme seront évalués. Deux étudiants d'un même binôme pourront avoir deux notes différentes en fonction de leur implication.