MASTER EN SCIENCES INFORMATIQUES À FINALITÉ SPÉCIALISÉE EN DATA SCIENCE

Est-ce que l'utilisation d'un jeu de rôle tangible peut aider à l'enseignement du paradigme orienté-objet ?

Wasterlain, Ludovic

Award date:
2020

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2019–2020

Est-ce que l'utilisation d'un jeu de rôle tangible peut aider à l'enseignement du paradigme orienté-objet ?

Ludovic Wasterlain

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Fanny Boraita

Co-promoteur : Tony Leclercq

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Remerciement

Je tiens à remercier ma promotrice Fanny Boraita, ainsi que mon co-promoteur Tony Leclercq pour leur patience, leur implication et leur dévouement dans ce mémoire. Leurs connaissances et leurs conseils m'ont permis de me dépasser et de ressortir plus grandi de cette expérience.

Je remercie Julie Henry pour son aide dans la création du support tangible. Son expérience dans le domaine m'a permis d'avancer rapidement. Je la remercie également de m'avoir partagé de nombreuses ressources sur le domaine de la tangibilité.

Je tiens également à remercier Paul Temple, Moussa Amrani, Victor Amaral de Sousa et Cédric Libert pour leur participation en tant qu'expert de l'orienté-objet.

Enfin, je tiens à remercier Julie Henry, Antoine Clarinval, Jérôme Maquoi, Maxime Dalla Valle, Boris Cherry et Mathieu Vandenneucker pour leur participation lors de l'expérimentation.

Table des matières

1 Introduction

Ce travail s'intéresse à deux thématiques : l'enseignement de l'orienté-objet et le tangible. L'enseignement de l'orienté-objet est un réel enjeu pour les enseignants du supérieur en informatique. Son utilisation récurrente dans les programmes actuels nécessite une compréhension claire et précise chez les apprenants. A côté, le caractère tangible d'un objet se définit comme sa capacité à être perceptible par le toucher. Nous manipulons donc tous les jours des outils tangibles, que ce soient nos smartphones, des jouets pour enfants ou encore des interfaces tangibles capables de résoudre une tâche bien précise. L'objectif est de créer des liens entre ces deux thématiques afin de proposer une nouvelle manière d'enseigner l'orienté-objet grâce à un support tangible.

Ce travail est divisé en deux grosses parties : un état de l'art et le développement d'une question de recherche. L'état de l'art est divisé en trois parties. La première se consacrera à l'enseignement de l'orienté-objet, la seconde se focalisera sur la tangibilité, tandis que la troisième présentera les liens entre l'orienté-objet et la tangibilité. Cet état de l'art présentera le contexte sur l'utilisation du tangible dans l'enseignement de l'orienté-objet.
Ensuite, ce travail présente une question de recherche à savoir la mise en place d'un nouveau dispositif pour l'enseignement de l'orienté-objet. Cette question est développée, une réponse est proposée, mesurée et discutée.
Finalement, une conclusion est dressée et des travaux futures proposés.

2 État de l'art

L'analyse de l'existant de ce travail se divise en trois parties. La première partie tâche de comprendre ce qui se fait actuellement en matière d'enseignement de l'orienté-objet. La seconde partie s'intéresse à l'utilisation du tangible dans différents domaines, notamment l'éducation et les jeux. La dernière partie concerne l'intégration de la tangibilité dans le domaine de l'enseignement de l'orienté-objet.

2.1 L'enseignement du paradigme Orienté-objet

2.1.1 Les étudiants concernés par l'enseignement de l'orienté-objet

Étant donné que ce sujet de recherche est lié à l'enseignement, il est important d'identifier quels sont les étudiants concernés par la programmation orienté-objet. Burton[1] apporte des éléments de réponses à cette question. D'un côté, les défenseurs de l'approche "objet en premier" disent que ce paradigme est devenu tellement important et tellement omniprésent qu'il est primordial pour les étudiants de le maîtriser le plus rapidement possible. Burton[1] défend l'idée qu'il faut d'abord enseigner la programmation procédurale avant de passer à la programmation orienté-objet. En effet, selon lui pour écrire correctement du code orienté-objet, il faut d'abord détenir les bases de la programmation (déclarations de variables, fonctions, boucles, ...). De plus, la modélisation orienté-objet demande un niveau d'abstraction supplémentaire qui peut mettre du temps à être compris par les étudiants. Enfin, la construction d'algorithmes efficaces requiert un niveau de concentration important. Intégrer des concepts orienté-objet en plus peut être déroutant pour les étudiants novices. Pour Burton[1], il faudrait donc diviser l'apprentissage, d'abord apprendre l'algorithmique, puis l'orienté-objet.

Cette idée est également soutenue par Decker [2]. Il recommande, en première année universitaire en informatique, l'apprentissage de la programmation procédurale et, en deuxième année, il préconise d'assimiler l'orienté-objet. Decker [2] soutient ceci en évoquant plusieurs raisons :

- le paradigme orienté-objet est un paradigme parmi d'autres. On peut se poser la question de savoir quel paradigme dominera les autres à l'avenir. Apprendre directement ce paradigme orienté-objet pourrait ainsi biaiser l'apprentissage d'autres paradigmes chez les étudiants ;
- l'apprentissage de l'orienté-objet serait trop complexe pour les étudiants de première année. Decker [2] explique que programmer de cette façon requiert un niveau d'abstraction et une vision informatique que les étudiants ne détiennent pas toujours lors de leur arrivée en première année. Au contraire, cette première année leur servirait justement à construire cette vision ;
- idéalement, il faudrait que tous les étudiants aient la même expérience et soient au même niveau avant d'apprendre l'orienté-objet. Les étudiants de première année sachant déjà programmer seraient déjà plus à l'aise avec les notions élémentaires de la programmation et ainsi avantagés par rapport aux étudiants complètement novices qui débutent en informatique ;
- dans l'orienté-objet, il y a beaucoup de formalisme et de concepts théoriques à comprendre. Decker [2] craint que les étudiants n'aient que trop peu d'occasions pour s'exercer à programmer ;
- la notion d'algorithme est trop importante en programmation. Peu importe le paradigme utilisé, il faut avant tout comprendre comment construire de beaux algorithmes efficaces ;
- les langages orienté-objet sont chargés, un étudiant débutant risque donc d'être vite perdu ;
- la première année d'un étudiant en informatique dans l'université du professeur Decker est fort chargée. Entre les cours de mathématiques nécessaires, celui d'introduction à la programmation et d'autres cours plus classiques, il y a peu de place pour un cours d'orienté-objet ;
- les cours d'orienté-objet rentrent bien dans le cursus de seconde année. Ici Decker parle

du point de vue de son université. En général, les étudiants en deuxième année d'informatique voient les structures de données, ils approfondissent leurs connaissances des algorithmes ou encore les bases de données. Ces différents domaines s'accordent bien avec l'enseignement de l'orienté-objet.

Pour ces raisons Decker recommande d'enseigner l'orienté-objet en seconde année d'informatique.

2.1.2 Approche d'enseignement de l'orienté-objet

Dans la littérature, plusieurs approches existent afin d'enseigner le paradigme orienté-objet. Ces différentes approches sont l'approche ascendante (*bottom-up*), l'approche descendante (*top-down*), l'approche constructiviste et l'approche par le jeu. Pour chacune de celles-ci, une définition va être donnée, suivie de son application dans le cadre de l'apprentissage de l'orienté-objet(OO).

Approche ascendante

L'approche ascendante dans l'enseignement consiste à étudier des éléments de base d'une discipline, puis de les assembler afin d'avoir une vision globale sur le sujet.

Dans le cas de l'OO, une telle approche signifie que le professeur va d'abord enseigner la syntaxe propre à un langage OO à ses étudiants. Ensuite, les étudiants vont voir les structures de données comme les piles, les arbres, les listes et comment fonctionne en détail le stockage de ces structures et enfin finir sur les différents concepts plus généraux de l'OO comme les classes ou les objets. Historiquement, cette approche prédomine dans l'enseignement de l'orienté-objet [3]. Néanmoins, Reek[4] met en évidence différents problèmes liés à cette approche. D'une part, apprendre des concepts abstraits aux étudiants peut être compliqué, car ces derniers vont d'abord penser à l'implémentation concrète sans prendre suffisamment de recul. D'autre part, si un cours de programmation, comme un cours d'introduction à la programmation, précède le cours d'OO, les étudiants rencontreront au début des notions qu'ils connaissent déjà. Cela peut donc créer chez certains étudiants la sensation qu'ils connaissent déjà tout ce qu'il y a à savoir sur le sujet. Cette sensation pourrait pousser certains étudiants à être moins attentifs au discours de leur professeur et ceux-ci risquent donc d'avoir des lacunes dans les concepts propres à l'OO.

Approche descendante

L'approche descendante est le contre-pied de l'approche précédente. En effet, elle consiste à d'abord définir les grandes lignes d'un sujet, puis de réduire progressivement jusqu'aux éléments de base.

Cette approche est recommandée par Reek[4] dans le cadre de l'enseignement OO. Les concepts et l'abstraction sont considérés comme les concepts primordiaux et doivent donc être vus en priorité. Une fois ces concepts maîtrisés par les étudiants, ils seront alors confrontés aux éléments de syntaxe et de structures de données propre au langage orienté-objet choisi par le professeur et pourront commencer des laboratoires plus pratiques. Une telle approche présente plusieurs avantages :

- le professeur peut confronter très tôt les étudiants à des structures de données plus complexes sans que ceux-ci ne doivent se soucier de l'implémentation qui se cache derrière ;
- les étudiants développent plus vite une meilleure vision pour résoudre des problèmes de taille raisonnable ;
- les étudiants ont plus facile à percevoir les programmes comme des collections de composants qui communiquent entre eux plutôt que comme des séquences d'instructions basiques ;
- les étudiants évitent de croire qu'écrire un programme signifie implémenter chaque partie de ce programme ;

— les étudiants sont rassurés sur le fait qu'ils n'ont pas besoin de savoir absolument comment toutes les parties du programme sont implémentées, particulièrement les parties venant de programme extérieur comme les librairies par exemple.

Cette idée de mettre les concepts en priorité au détriment du code est mise en avant par Sims-Knight [5], qui démontre qu'apprendre la programmation orientée-objet sans devoir programmer est tout à fait possible. Selon lui, les étudiants peuvent parvenir à apprendre la programmation en regardant un expert modéliser un phénomène puis en modélisant eux-mêmes un phénomène similaire en s'appuyant sur des concepts et des notions apportés par l'expert. L'expert montre ensuite l'héritage sur le premier phénomène et les étudiants doivent alors le déterminer sur le second. Une fois ces concepts acquis par les étudiants, ils passent à l'étape programmation.

Approche constructiviste

L'approche constructiviste met en avant l'expérience des étudiants et leur fournit toutes les ressources nécessaires afin qu'ils puissent construire eux-mêmes leurs connaissances. Dans le cadre de l'enseignement de l'OO, l'approche constructiviste définit par Hadjerrouit [6] consiste en un cadre de travail à suivre pour enseigner l'OO. Ce cadre présente d'abord les trois types de savoirs à enseigner dans le domaine de l'orienté-objet :

1. les différents concepts qui gouvernent le paradigme orienté-objet tels que les classes, les objets, l'abstraction, l'encapsulation, le polymorphisme et la modularité ;
2. le langage de programmation orienté-objet, comme JAVA ou C++, permettant d'implémenter ces concepts ;
3. les problèmes spécifiques à résoudre qui permettent de mettre en valeur les concepts orienté-objets.

A côté, six principes sont définis pour encadrer l'enseignement :

— les concepts orienté-objets doivent être activement construits par l'étudiant, pas simplement transmis par le professeur ;
— le développement de programmes, même les plus petits programmes, doivent être guidés par ces concepts ;
— une évaluation des connaissances en programmation de l'étudiant est nécessaire pour voir si elles entrent en conflit avec les concepts. Si une anomalie ou une représentation erronée est détectée, cela permet d'y remédier le plus tôt possible et d'éviter que cela empire ;
— les différents concepts ont besoin de liens forts avec les problèmes situationnels et le langage de programmation afin de faciliter la résolution de problèmes ;
— les cours classiques, type ex cathedra, doivent être remplacés par un ensemble d'activités pour que l'étudiant puisse pratiquer et donc construire ses connaissances ;
— les problèmes illustratifs doivent être suffisamment réalistes pour garder l'attention de l'étudiant et ainsi le garder motivé ;

L'auteur précise que ce cadre a besoin d'être testé sur le long terme afin d'être validé. Une critique qui peut être adressée à ce travail est que, même si cette approche constructiviste cadre bien l'apprentissage de l'orienté-objet, elle requiert une participation active des étudiants afin d'être optimale.

Approche par le jeu

La dernière approche présentée dans ce travail est l'approche par le jeu, notamment grâce à des *Serious Games* et des jeux éducatifs plus classiques.

L'encyclopédie Universalis définit le jeu comme : "Une activité qui semble échapper aux normes de la vie sociale. Jouer, c'est se situer en dehors des contraintes qui régissent l'existence." Cinq critères sont nécessaires pour définir le jeu [7] :

1. la fiction "réelle", le second degré : le jeu prend place dans un cadre spatio-temporel donné et implique une métacommunication. Le joueur s'y investit avec autant de sérieux que dans la réalité ;

2. la décision du joueur : un jeu implique des prises de décisions de la part du joueur ;
3. la règle : le jeu est structuré par des règles et ces règles sont acceptées par l'ensemble des joueurs ;
4. la frivolité : le jeu n'a aucune conséquence sur la réalité ;
5. l'incertitude : c'est le moteur de jeu. Le jeu n'est jamais deux fois pareil. On ne sait jamais à l'avance comment il va se dérouler.

Quant aux *Serious Games*, ils sont définis comme : "Des applications de simulation/formation qui utilisent les dernières technologies issues du monde du jeu vidéo et de la réalité virtuelle." Cependant, cette définition disparaît progressivement, tout en conservant un lien fort avec le jeu vidéo, afin d'inclure des notions de jeu avec un double aspect : soit le jeu est déconnecté du contenu à apprendre, soit l'objet même du jeu est le contenu à acquérir. En d'autres termes, soit le joueur joue à un jeu et le contenu éducatif auquel il accède n'est pas en rapport avec le jeu, soit il accède à ce contenu éducatif de manière traditionnelle et claire.[8].

L'approche par le jeu consiste à intégrer ces outils ludiques dans un cursus d'enseignement. L'intégration peut se faire de différentes façons. En effet, le jeu peut servir pour illustrer certains concepts ou encore, le jeu peut mettre en application ces concepts. L'approche par le jeu ne se limite pas aux jeux vidéo et aux *Serious Games*. En effet, le jeu peut prendre différentes autres formes, que ce soit des jeux de plateau, des jeux de rôle ou encore des jeux dit tangibles tant que cette activité correspond à la définition de jeu donnée précédemment.

Maintenant qu'une définition du jeu a été donnée, deux questions restent en suspens : est-ce que l'approche par le jeu est faisable dans l'enseignement de l'orienté-objet ? Quels sont les jeux éducatifs sur l'orienté-objet existants ?

Rais [9] démontre la faisabilité et l'utilisabilité de l'enseignement de l'orienté-objet par le jeu. Selon lui, par rapport aux autres approches, l'approche par le jeu permet une meilleure compréhension des concepts OO ciblés par le jeu. Un jeu n'arrive pas à cibler tous les concepts et n'est donc pas complet, des concepts fondamentaux sont parfois oubliés. L'approche par le jeu est également plus amusante que les autres approches et donnerait plus envie d'apprendre.

Pour ce qui est des jeux traitant de l'orienté-objet, une étude réalisée par Abbasi [10] sur toute une série de jeux, met en lumière trois façons d'utiliser le jeu dans l'enseignement de l'OO : apprendre en jouant au jeu, apprendre en créant le jeu et apprendre en utilisant d'autres outils autour du jeu. Il existe également plusieurs façons d'inclure l'OO dans le jeu : faire un jeu où les différents concepts sont au centre du jeu, faire un jeu où il faut d'abord programmer, ou créer un jeu qui possède une interface utilisateur graphique avec laquelle l'utilisateur doit interagir.

Voici divers exemples de jeux traitant de l'orienté-objet, un tableau récapitulatif de ces jeux se trouve à la fin de cette section.

Le premier jeu, est Alice [1] [11] [12].Il s'agit d'un programme permettant aux étudiants de raconter des histoires et de les mettre en scène. Le fonctionnement est très simple et similaire à Scratch [2]. Le jeu invite l'utilisateur à créer un programme grâce à des briques de programmation (voir Figure 1). Lorsque le joueur compile son programme, les briques s'exécutent les unes après les autres. Parmi ces briques, il y a des éléments classiques de programmation comme des boucles, des conditionnels,... Mais il y a aussi des briques plus spécifiques au paradigme orienté-objet. Une partie importante d'Alice est la gestion des objets : tous les éléments qui représentent les acteurs, les décors, les accessoires et autres représentations du monde réel sont des objets possédant des méthodes, des procédures, des variables. La caméra elle-même est un objet qui possède des méthodes pour faire des zooms, des rotations, etc. Dans le cadre de l'étude réalisée par Kelleher[12], Alice a été utilisé par des élèves âgés de 11 à 15 ans afin d'encourager les jeunes de cet âge à entreprendre des études en informatique. Alice permet aux étudiants de comprendre

1. http://www.alice.org/
2. https://scratch.mit.edu/

les bases de l'orienté-objet. Son gros point fort est l'attractivité auprès des étudiants, ceux-ci ont démontré un intérêt pour ce jeu et ils ont manifesté une envie d'y rejouer même en dehors des heures de cours. Il a été observé également que les étudiants arrivaient facilement à débugger leurs programmes et à détecter rapidement ce qui n'allait pas.

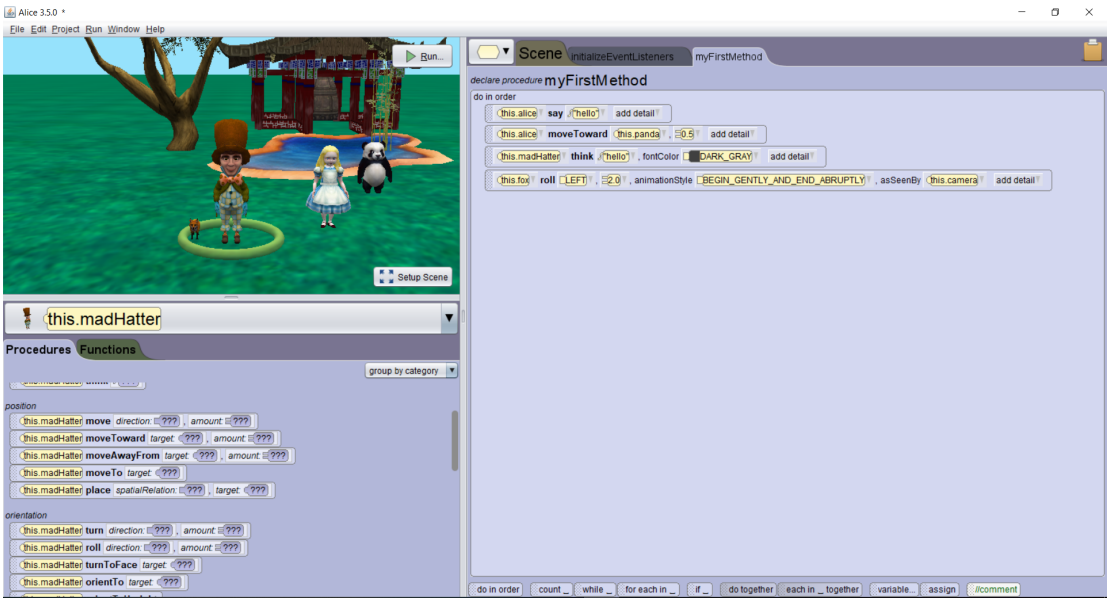


FIGURE 1 – Alice 3

Le second jeu est Greenfoot [13]. Il s'agit d'un environnement qui permet de créer des simulations interactives. Les acteurs du monde sont des objets qu'on peut placer directement dans le monde du jeu et les utilisateurs peuvent jouer avec ces acteurs en invoquant différentes méthodes sur ceux-ci (voir Figure 2). Greenfoot met principalement l'accent sur le concept d'objet, souvent mal compris par les étudiants. La différence avec Alice est que l'utilisateur ne manipule pas des briques de programmation, il doit lui-même implémenter le comportement des différents acteurs en JAVA. Il existe une documentation en ligne sur ce que l'utilisateur peut faire et Greenfoot utilise seulement une version simplifiée de JAVA afin de correspondre aux besoins des joueurs. Une fois les différentes classes des acteurs définies, l'utilisateur peut alors cliquer sur les différents acteurs et utiliser les différentes méthodes ou utiliser une fonction *run* qui permet de définir une séquence de programmation puis de voir le résultat. L'avantage est que l'utilisateur voit directement sur une interface graphique l'effet de la méthode qu'il vient d'invoquer. Greenfoot est régulièrement accompagné de BlueJ. BlueJ [3] est un environnement de développement qui supporte l'apprentissage et l'enseignement de la programmation orienté-objet. L'interface de BlueJ est similaire à Greenfoot, la principale différence est que Greenfoot utilise des images pour représenter ses acteurs, tandis que BlueJ utilise une représentation qui se rapproche des diagrammes de classes. Comme avec Greenfoot, l'utilisateur peut invoquer directement les méthodes sur les différents objets en un simple clic. Yan[14] a démontré l'utilité de Greenfoot et BlueJ dans l'enseignement de la programmation orienté-objet. Dans le cadre de cette étude, les étudiants apprenant l'orienté-objet devaient utiliser Greenfoot et BlueJ. Cette étude à permis de mettre en valeur les bienfaits de Greenfoot comme premier pas dans l'orienté-objet. En effet, il permet d'améliorer l'intérêt de l'étudiant et de renforcer les concepts orienté-objet, car les exemples sont réels et visuels.

3. https://www.bluej.org/

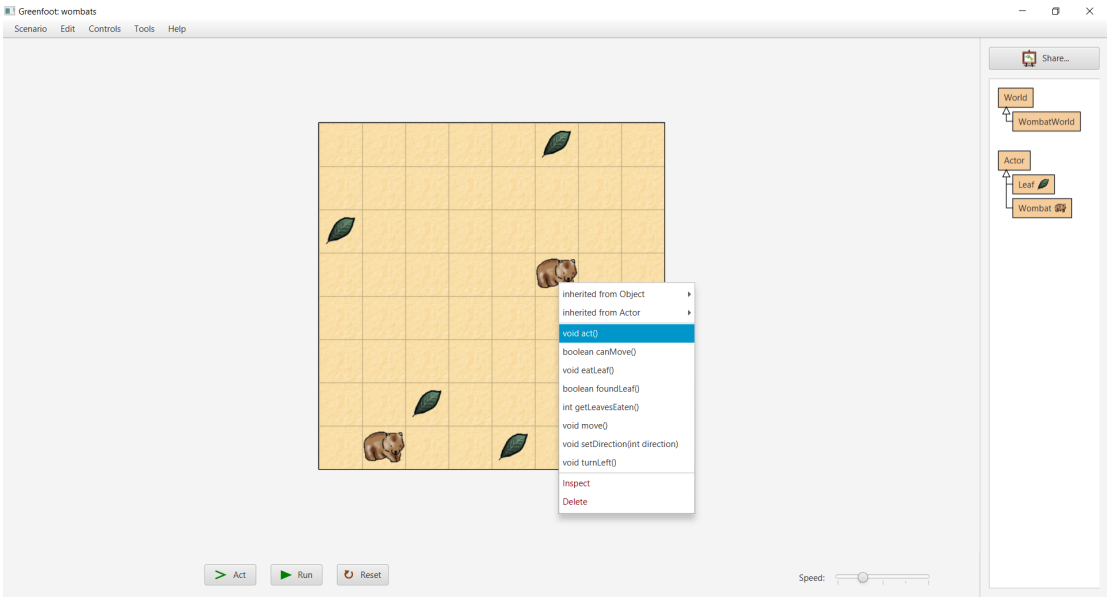


FIGURE 2 – Greenfoot

Le prochain jeu de Sharma[15] consiste en trois *sous-jeux*. Il a utilisé ces jeux dans le cadre d'un cours sur l'orienté-objet afin de les faire tester par les étudiants. Chacun de ces jeux se concentre sur un concept particulier du paradigme orienté-objet. Le premier se concentre sur l'héritage, le second sur le polymorphisme, tandis que le troisième se consacre à l'encapsulation. Les différents jeux consistent en une série de questionnaires accompagnés d'une illustration en 3D représentant la situation. Les retours sur ces jeux montrent que les différents jeux ont eu un impact non seulement sur la compréhension des différents concepts abordés par les modules, mais également sur leur intérêt pour le sujet.

Le dernier jeu présenté ici est un jeu mobile intitulé *Odyssey of Phoenix*[16]. Il s'agit d'un jeu de rôle qui incorpore les différents concepts orienté-objet au coeur même du jeu. Le jeu raconte l'histoire d'un aventurier qui veut construire un vaisseau spatial. Pour ce faire, il doit récolter des ressources afin de construire les différentes parties de son véhicule. Le joueur dispose d'un livre de conseils pour lui montrer les ressources dont il a besoin pour construire les différentes parties. Ces conseils représentent le concept d'encapsulation des classes et des objets. Le joueur dispose également du pseudo-code du processus de construction des parties du vaisseau (voir Figure 3). Le concept d'héritage est également présent car des parties similaires du vaisseau sont liées par des relations parents-enfants, par exemple le moteur avant et le moteur arrière sont tous les deux issus de la classe moteur et donc ils ont besoin des mêmes ressources pour être fabriqués. Les étudiants ont montré un vrai intérêt pour ce jeu. Avant de tester le jeu, les étudiants ont répondu à un questionnaire sur différentes notions orienté-objet. Une fois le jeu testé, ils ont du répondre à un nouveau questionnaire. Globalement, une hausse des résultats du questionnaire a été observée. Les étudiants ont affirmé avoir passé un bon moment à jouer au jeu et cela les a encouragés à apprendre la programmation orienté-objet. Un dernier point important mis en évidence par ce jeu est qu'il est préférable d'utiliser du pseudo-code plutôt que du code lorsqu'on développe un jeu éducatif. En effet, cela permet aux étudiants de s'abstraire des notions de syntaxe et donc de se focaliser sur le comportement des procédures et des méthodes.

FIGURE 3 – Odyssée du Phoenix - Classes

Le Tableau 1 suivant résume les différents jeux présentés dans cette section

Nom du jeu	**Concepts OO**	**Inclusion des concepts dans le jeu**	**Résultats des tests**
Alice[11][12]	Classe, Objet, Méthode, Héritage	L'utilisateur doit interagir avec les concepts via l'interface graphique du jeu	Permet une compréhension des concepts de bases de l'OO, attractif, augmente l'intérêt pour l'OO et la programmation, amusant.
Greenfoot et BlueJ[13][14]	Classe, Objet, Méthode	Les notions de classe et d'objet sont introduites, puis il faut programmer les méthodes	Renforce la compréhension des concepts OO présents dans le jeu, augmente l'intérêt pour l'OO.
Sharma[15]	Héritage, Encapsulation, Polymorphisme	L'utilisateur doit répondre à différents questionnaires en lien avec les concepts	Renforce la compréhension des concepts OO présents dans le jeu, augmente l'intérêt pour l'OO.
Odyssey of Phoenix[16]	Classe, Objet, Méthode, Héritage	Les différents concepts sont au cœur du jeu	Renforce la compréhension des concepts OO présents dans le jeu, amusant, augmente l'intérêt pour l'OO, la compréhension du code n'est pas un frein car le jeu utilise du pseudo-code

TABLE 1 – Les différents jeux

La plupart des jeux présentés abordent des concepts basiques et essentiels du paradigme. Ceux-ci sont utilisés pour introduire l'OO aux étudiants. Contrairement au jeu de Sharma qui s'adresse à des étudiants ayant déjà des notions en OO. Ce jeu se permet donc d'aborder des concepts plus précis, qui requièrent une connaissance de base de l'OO. Les résultats observés sur l'ensemble des jeux sont similaires. Ceux-ci permettent de renforcer la compréhension des concepts qu'ils abordent et ils augmentent globalement l'intérêt de l'OO auprès des joueurs. Ces résultats sont également similaires à ceux de Rais[9] présentés précédemment. Selon lui, les jeux sont efficaces pour améliorer la compréhension des concepts qu'ils ciblent, mais ne permettent pas de cibler efficacement l'ensemble des concepts. Les jeux doivent se limiter à un nombre restreint de concepts.

Comparaison des différentes approches

Le Tableau 2 résumant les différentes approches présentées dans ce travail.

Approche	Principe	Résultats observés
Descendante	D'abord étudier les éléments syntaxiques d'un langage et les structures de données. Puis étudier les concepts.	Différents problèmes sont observés : les étudiants penseront d'abord à l'implémentation avant de penser aux concepts, peut donner la sensation de tout savoir chez les étudiants ayant déjà des bases en programmation.
Ascendante	D'abord étudier les concepts, puis réduire progressivement jusqu'à étudier les éléments syntaxiques et les structures de données.	Les étudiants développent une meilleure vision pour résoudre des problèmes, le professeur peut montrer des structures de données complexes aux étudiants sans se focaliser sur leur implémentation, les étudiants se concentrent sur les concepts plutôt que sur le code.
Construcitiviste	L'étudiant doit construire tout seul ses connaissances. L'étudiant doit comprendre les concepts puis résoudre des problèmes spécifiques pour améliorer sa compréhension de ces concepts.	Aucun résultat sur le long terme pour l'instant. Cette approche requiert une participation active des étudiants.
Jeu	Utiliser des outils ludiques dans l'enseignement de l'OO. Le jeu doit illustrer les concepts OO ou les mettre en application.	Augmente l'intérêt de l'orienté-objet chez les étudiants, améliore la compréhension des concepts ciblés par le jeux. Cependant, les jeux ne permettent pas de cibler l'ensemble des concepts, ils sont généralement incomplets.

TABLE 2 – Les différentes approches

Globalement, l'approche descendante est à éviter dans le cadre de l'orienté-objet. L'approche ascendante est encouragée, tandis que l'approche constructivste n'a pas encore de résultat validé sur le long terme pour tirer des conclusions. Quant à l'approche par le jeu, sa capacité à intéresser les étudiants n'est pas négligeable. Son principal problème est de ne pas être assez complet car il est difficile d'inclure l'ensemble des concepts OO dans un seul et même jeux. Une piste pour pallier à ce problème serait d'inclure cette approche avec une autre, l'ascendante par exemple, afin de coupler les avantages de ces deux approches.

2.1.3 Problèmes fréquents rencontrés par les étudiants apprenant l'orienté-objet

Plusieurs problèmes peuvent survenir chez les étudiants lors de l'apprentissage du paradigme orienté-objet. Ces problèmes peuvent prendre différentes formes, comme l'incompréhension d'un énoncé ou d'un concept ou encore détenir des représentations erronées.

Garner [17] liste les différents problèmes rencontrés par ses étudiants en informatique au cours de différents laboratoires effectués. Les étudiants de cette étude suivaient tous un cours de JAVA. Les données ont été récoltées suite à des entretiens avec des étudiants montrant différentes lacunes en programmation et des problèmes sur les concepts de programmation au sens large. Les résultats de cette étude montrent trois raisons principales pour lesquelles les étudiants de cette étude ont du mal à programmer.

1. Les étudiants comprennent le problème qui leur est demandé de résoudre et arrivent à trouver une solution, mais ils n'arrivent pas à traduire cela en un algorithme, ils ne savent pas comment commencer par exemple ;
2. Les étudiants ont du mal avec les structures de bases. Ils comprennent le design des classes, mais ils n'arrivent pas à appréhender les détails. Par exemple, ils écrivent du code en dehors des méthodes, ou bien ils définissent des variables en dehors des classes ;
3. Les étudiants ne comprennent pas la tâche qui leur est demandée ou ils ne comprennent pas la solution.

D'autres raisons, moins récurrentes sont également évoquées. Cependant, ces problèmes moins fréquents ont tendance à s'effacer "naturellement" au cours des laboratoires tandis que les problèmes plus fréquents sont beaucoup plus tenaces et ont donc du mal à partir :

— ils rencontrent des problèmes avec les outils, ils n'arrivent pas à utiliser correctement ou à comprendre l'environnement de développement, ils se perdent dans les différents fichiers présents sur leur machine ;

— ils ont du mal à nommer les choses. Ils se retrouvent avec des variables x, y ou z, ou encore des méthodes avec des noms peu évocateurs ;
— ils ont des problèmes avec les détails, notamment les détails de syntaxe. Ils oublient des points-virgules, de fermer leurs accolades, leurs parenthèses ;
— les exceptions et les *catchs* sont généralement mal compris ;

Un autre problème fréquent avec l'enseignement de l'OO est que les étudiants apprenant ce paradigme peuvent avoir des représentations erronées. Ces représentations, souvent bien ancrées chez les étudiants, peuvent rendre l'apprentissage correct et efficace de l'orienté-objet relativement long et complexe. Une mauvaise interprétation des concepts par les étudiants peut prendre du temps à être corrigée et induire l'étudiant en erreur. Les étudiants rencontrent souvent les mêmes difficultés. Une liste des erreurs de compréhension et des représentations erronées les plus fréquentes chez les étudiants apprenant l'orienté-objet au niveau universitaire a été créée par Ragonis et al.[18]. Les données de cette liste ont été obtenues sur base d'une série de questionnaires et d'interviews auprès d'étudiants de seconde année en informatique ayant suivi un cours d'orienté-objet. Cette liste est divisée en quatre parties : les objets et les classes, l'instanciation et la construction, les classes simples et les classes composées et enfin, l'exécution d'un programme. Ces quatre parties sont détaillées ci-dessous.

Objets vs Class

Les étudiants ont du mal à comprendre la nature d'une classe en tant que template. En effet, ils éprouvent des difficultés à comprendre la nature statique d'une classe mais également à comprendre qu'une méthode peut être invoquée sur n'importe quel objet de cette classe. Ils pensent qu'on ne peut invoquer une méthode qu'une seule fois ou encore qu'on peut ajouter une méthode qui ajoute des attributs à la classe ou qu'on peut définir une méthode qui n'a accès à aucun attribut.

Plus précisément, au niveau de leurs représentations, il ressort de cette étude que les étudiants ont du mal à comprendre la relation entre un objet et une classe :
— ils se représentent une classe comme étant une collection d'objets ;
— ils croient qu'ils peuvent définir une méthode non constructeur pour créer un nouvel objet ;
— ils croient qu'ils peuvent définir une méthode qui remplace l'objet lui-même, voire même le supprimer ;
— ils croient qu'ils peuvent définir une méthode qui va séparer un objet en deux nouveaux objets.

Enfin, ils ont des difficultés à identifier les objets. Selon ces étudiants, deux objets d'une même classe ne peuvent pas avoir les mêmes attributs ou que deux objets peuvent avoir le même identifiant s'il n'y a pas de différences dans leurs attributs.

Un revue de littérature, cette fois-ci menée par Xinogalos[19], cherche également à comprendre et catégoriser ces représentations erronées en se concentrant seulement sur les conceptions d'objet et de classe. Xinogalos commence en présentant une série de représentations erronées récurrentes chez les étudiants :
— les étudiants ont tendance à confondre objet et classe mais principalement pour les classes définies par l'utilisateur, pas pour les classes issues de libraire ;
— les objets sont vus comme des conteneurs de variables, ce qui implique que toutes les variables d'instances contenues dans un objet doivent être du même type ;
— les objets sont vus comme un enregistrement d'une base de données, ce qui néglige complètement l'aspect comportemental d'un objet ;
— une classe est vue comme une collection d'objets plutôt que comme une abstraction, un template afin de construire des objets ;
— la relation classe/objet est perçue comme l'équivalent de la relation ensemble/sous-ensemble ;
— les étudiants perçoivent un objet comme étant une partie d'une classe.

En plus de ces représentations erronées, il cite différentes erreurs de compréhensions récurrentes chez ses étudiants :

— définir plusieurs classes peut être compliqué pour les étudiants. Les étudiants sont perdus car ils ne savent pas s'ils doivent mettre toutes les classes dans le même fichier ou bien faire un fichier par classe ;
— les classes composées sont généralement mal comprises, définir une classe avec des attributs appartenant à d'autres classes est compliqué pour les étudiants ;
— ils ont également des problèmes à modéliser des classes et les relations entre elles, ils ne comprennent pas qu'une classe modélise un phénomène du monde réel ;
— les étudiants voient les objets comme des entités statiques, ils ne voient pas l'aspect dynamique et l'aspect comportemental d'un objet.

La revue de littérature de Xinalogos met en évidence la fréquence de ces représentations erronées afin de déterminer lesquelles sont les plus communes. Les résultats sont les suivants : globalement, le concept de classe est moins souvent sujet à des représentations erronées que le concept d'objet. En effet, là où 69% des étudiants se représentent correctement la nature d'une classe, comme étant un modèle décrivant un type d'objet, seulement 50% des étudiants se représentent correctement le concept d'objet, comme étant une entité d'un phénomène du monde réel.

Instanciation et construction

Les étudiants ont une mauvaise compréhension des concepts d'instanciation et de construction d'objets. En effet, ils pensent que ce n'est pas nécessaire d'invoquer un constructeur lorsqu'on veut créer un nouvel objet car sa définition est suffisante. Ils pensent également que les constructeurs ne peuvent contenir que de l'assignation de variables.

L'instanciation d'un objet dans une classe composée peut également leur poser des problèmes. Ils croient que si un objet d'une classe simple existe, il n'est pas nécessaire de créer un objet de la classe composée. Inversement, ils croient que la création d'une classe composée crée automatiquement l'objet de la classe simple qui apparaît comme attribut de la classe composée.

Classe simple vs Classe composée

Les étudiants comprennent mal la notion d'encapsulation. En effet, toute une série de représentations erronées sont liées à ce concept.

— ils croient qu'un objet ne peut pas être la valeur d'un attribut ;
— ils pensent que les attributs de la classe composée incluent tous les attributs de la classe simple plutôt que l'objet en tant que tel ;
— ils pensent que les méthodes de la classe simple doivent être déclarées également dans la classe composée pour chaque objet simple ;
— ils pensent que pour changer la valeur d'un attribut d'un objet de classe simple dans un objet de classe composée, il faut créer un nouvel objet ;
— ils pensent que des méthodes ne peuvent être invoquées que sur des objets de classes composées et pas sur des objets de classes simples qui sont donc des attributs.

La notion de modularité entre les classes composées et les classes simples est problématique également. Les méthodes des classes simples sont rarement utilisées. A la place, des méthodes équivalentes sont créées et définies dans la classe composée. En plus de cela, les étudiants ont du mal à différencier les méthodes de classes différentes si elles ont le même nom. Les étudiants croient également qu'une fois une classe composée définie, il n'est plus possible d'ajouter de méthodes dans la classe simple.

Les étudiants voient les classes comme des collections d'objets, ce qui est problématique. Donc ils pensent que les objets utilisés comme valeur d'attribut de la classe composée doivent être identiques. Ils pensent également que dans une classe composée, on peut définir une méthode qui ajoute ou qui supprime des attributs de la classe simple.

L'exécution du programme

Les étudiants ont des difficultés à comprendre l'exécution des méthodes. Ceux-ci pensent que les méthodes sont exécutées en fonction de l'ordre dans lequel elles ont été définies, ou bien que les méthodes ne peuvent être utilisées qu'une seule fois.

Maintenant que ces erreurs de compréhension et représentations erronées ont été listées, comment les enseignants peuvent-ils faire afin de les éviter au maximum ?

En ce qui concerne les représentations sur les objets et les classes, la revue de littérature de Xinalogos [19] propose des pistes de réflexions pour éviter au maximum ces problèmes récurrents. Il faut présenter les classes et les objets de manière claire et concise, un objet est une entité qui représente un phénomène du monde réel tandis qu'une classe est un modèle décrivant des types d'objets. L'article préconise également les activités manuelles plutôt que de demander du code directement, et donc d'exposer graduellement les étudiants au code.

Holland [20] propose des pistes de réflexions afin d'éviter le plus possible les problèmes cités auparavant. Selon lui, il est facile d'éviter les plus grosses erreurs de compréhension en utilisant des exemples introductifs pertinents.

Dans le cas de la représentation erronée suivante :"Les objets ne sont pas qu'un stockage de données", il estime que pour ne pas oublier l'aspect comportemental de l'objet dans ce cas, il faut utiliser une classe qui n'aura pas le même comportement en fonction de l'état d'un objet. Par exemple, utiliser un objet de la classe *Account* qui représente le compte en banque d'une personne, son état et donc son comportement seront différents en fonction du montant sur son compte. Si cette personne essaye de retirer des sous de son compte alors qu'il a un solde négatif, l'objet se comportera d'une certaine façon, en bloquant l'opération par exemple. Tandis que si la personne possède suffisamment d'argent, l'opération sera validée. On remarque bien que les comportements diffèrent en fonction de l'état de l'objet.

Une autre erreur grossière et fréquente des étudiants qui peut facilement être évitée est la confusion entre le concept de classe et le concept d'objet. Pour ce faire, Holland suggère de faire en sorte que tous les objets utilisés en guise d'exemple soient issus d'une seule classe avec plusieurs instances. Si on reprend l'exemple du compte en banque, il faut alors utiliser plusieurs comptes différents, en variant bien les attributs de ces objets pour bien montrer qu'ils sont différents.

Ensuite, une erreur qui peut être facilement évitée est la confusion entre un objet et une variable. Pour y remédier, il faut utiliser des objets qui possèdent plus d'une variable d'instance. Pour rester sur l'exemple du compte en banque, il faut qu'il y ait plusieurs variables d'instance, nom du client, montant, âge du client...

Enfin, Holland recommande des bonnes pratiques à adopter lors de l'enseignement de l'orienté-objet afin de minimiser le risque de représentations erronées et d'erreur de compréhension :

- faire en sorte que les étudiants assignent un objet à plusieurs variables, démontrer que chaque variable référence le même objet et montrer que lorsque l'objet change d'état, cela se répercute sur toutes les variables ;
- avoir deux variables qui référencent le même objet, puis réassigner une des deux variables à un nouvel objet et bien montrer que l'autre variable référence toujours l'objet initial ;
- il faut faire du *swapping*, échanger les variables qui référencent deux objets différents en utilisant une variable intermédiaire ;
- montrer que deux instances différentes peuvent avoir le même état ;
- montrer que deux objets qui ont le même état à un moment donné ne sont pas le même objet en envoyant des messages pour faire diverger leur état.

2.2 Tangibilité

Après avoir présenté l'enseignement de l'orienté-objet, la suite de ce travail va se consacrer à la tangibilité. En effet, afin de voir quels liens existent entre orienté-objet et tangibilité, il est important de définir la tangibilité d'un point de vue théorique. Cette approche théorique se fera par le prisme des interfaces utilisateurs tangibles. Ensuite, différents exemples d'utilisations concrètes dans le domaine de l'éducation puis dans le domaine des jeux tangibles seront présentés.

2.2.1 La tangibilité et les interfaces utilisateur tangible

Les différentes notions liées à la tangibilité sont présentées par le prisme des interfaces utilisateurs tangibles. En effet, ces différents outils sont fort répandus dans le domaine de la tangibilité et permettent d'aborder facilement et concrètement la thématique.

Le terme tangible est défini par le Centre National de Ressources Textuelles et lexicales [4] comme "Ce qui est perceptible par le toucher". Le tangible reprend donc tous les éléments matériels qui sont concrets, manipulables.

Les interfaces utilisateurs tangibles, ou *Tangible User Interface* (TUI) en anglais sont des interfaces utilisateurs où l'utilisateur interagit avec de l'information digitale à travers un environnement physique. Cette première définition date de 1996 et est donnée par Fitzmaurice[21]. Il propose un nouveau type d'interface utilisateur, qui est fondamentalement différent du type d'interface classique, à savoir les *Graphical User Interface*, ou GUI. Il nomme sa nouvelle idée *Graspable user interface*, une interface utilisateur qu'on peut saisir, manipuler. Les travaux de Fitzmaurice sont ensuite repris par Ishii [22]. Dans son article, celui-ci présente ce nouveau type d'interaction homme-machine, qui contient du son, du touchable ou encore du physique. La définition de Fitzmaurice est reprise par Markova [23] qui synthétise les différents travaux précédents afin d'avoir une définition commune à toutes les interfaces tangibles. Une interface utilisateur tangible est donc une extension du monde digital qui est touchable et manipulable physiquement, et dont la manipulation a un impact sur l'état du monde digital. Quatre critères doivent être respectés afin de correspondre à une interface tangible :

1. objets tangibles : les TUI doivent posséder un ou plusieurs objets tangibles qui sont utilisés comme *input* et éventuellement comme *output*. Cela veut dire qu'une TUI peut se servir d'objets tangibles comme *input* et l'*output* est fourni séparément, sur un écran par exemple ;
2. incarnation : les TUI sont des systèmes où l'*input* et l'*output* doivent être spatialement proches et temporellement liés. Cela veut dire que lorsqu'un utilisateur entre un *input*, l'*output* produit doit apparaître dans son voisinage proche, que ce soit sur un objet tangible ou sur un écran ;
3. métaphore : il doit y avoir une analogie entre le monde digital et le monde réel. Les objets et actions du monde réel doivent correspondre à leur représentation du monde digital ;
4. continuité : un utilisateur ne cesse pas d'interagir avec une TUI. L'interface invite toujours l'utilisateur à interagir avec elle.

Ces interfaces tangibles peuvent prendre vie sous différentes formes. Shaer[24] définit 4 différents types d'interfaces tangibles possible.

Le premier est la réalité augmentée tangible. Il s'agit d'une combinaison d'*input* tangible avec un affichage des *outputs* en réalité augmentée.

Le troisième type est l'affichage ambiant. L'affichage ambiant permet d'accéder à des informations particulières, sur un écran par exemple. Avec le tangible, l'information se retrouve sur un objet avec lequel l'utilisateur peut interagir. Par exemple, une fleur qui fleurit lorsqu'un collègue est disponible pour un rendez-vous [25].

4. https://www.cnrtl.fr/lexicographie/Tangible

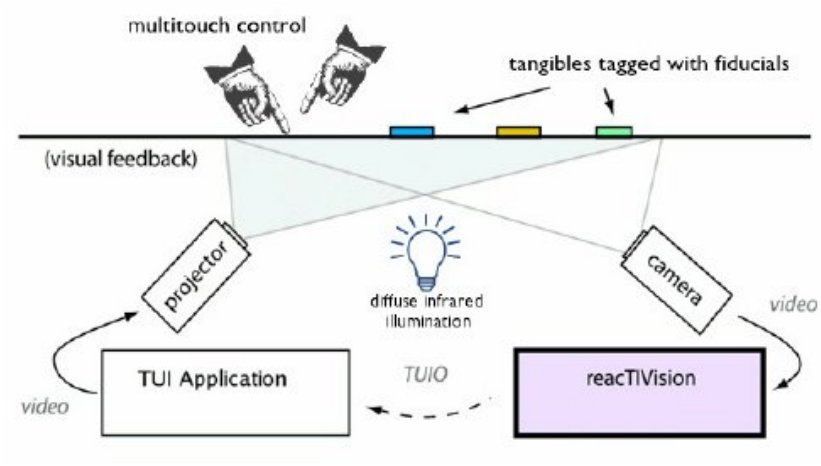


FIGURE 4 – Architecture d'une tabletop

Le dernier type est les interfaces utilisateurs intégrées. Dans ce cas, un objet virtuel est attaché à un objet physique que l'utilisateur peut manipuler, comme par exemple une figurine.

Ces différents types d'interfaces sont utilisés dans des domaines applicatifs variés. Ces interfaces sont présentes dans l'éducation (ce domaine sera expliqué en détail dans la sous-sous-section 2.2.2), dans l'urbanisme, dans la visualisation d'information, dans le divertissement ou encore dans les interactions sociales. Chacun des différents domaines présentés ci-dessus sera expliqué, accompagné d'un exemple d'application réelle.

La tangibilité dans l'urbanisme

Les TUI peuvent être utiles lorsqu'il s'agit de trouver une solution à un problème bien précis. En effet, les TUI permettent de faciliter le travail mental et permettent donc de trouver des solutions rapidement. Voici quelques exemples :

URP [26] est une table lumineuse interactive utilisée dans le domaine de la planification et le design d'espace urbain. Elle permet de poser des modélisations d'architecture physique sur une table et de voir les ombres projetées sur la table en fonction du moment de la journée, voir Figure 5.

MouseHaus Table [27] est une table interactive pour désigner collaborativement un espace urbain. Les utilisateurs placent sur la table différents blocs de couleurs représentant différents types d'espaces urbains (le vert est un parc,...). Une fois les espaces urbains placés, un programme simule le comportement des piétons afin de voir comment ils réagissent en fonction des types d'espaces qu'ils ont à leur disposition, voir Figure 6.

La tangibilié pour visualiser de l'information

La visualisation de l'information est un domaine informatique dont le but est de représenter visuellement des données sur une interface graphique. La visualisation de l'information est une tâche complexe qui demande beaucoup d'interactions avec l'utilisateur. Les TUI et la manipulation de celles-ci avec deux mains dans le monde réel offrent énormément de possibilités de visualisation.

Un exemple d'application utilisé pour cette discipline est GeoTUI [28] . Il s'agit d'une *tabletop* tangible destinée aux géophysiciens. Un modèle 3D est présent sur la table et les utilisateurs peuvent effectuer des plans de coupes afin de voir et d'analyser les différentes couches du sol. Pour effectuer des coupes, il suffit de déplacer deux palets sur la table afin de créer une ligne de coupe. Un boîtier avec des boutons permet de sélectionner la coupe désirée.

FIGURE 5 – URP, une table lumineuse pour le design de l'urbanisme urbain

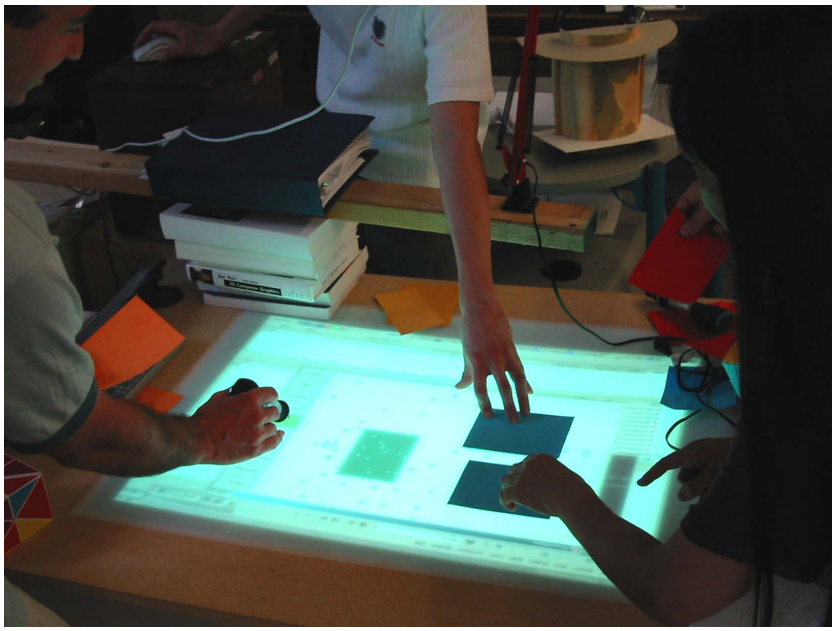


FIGURE 6 – MouseHaus

La tangibilité dans le divertissement

Le domaine du divertissement est un terrain très fertile pour l'utilisation d'interfaces tangibles, principalement à travers le biais des jeux tangibles (plus de détails dans la partie 2.2.3). Mais il n'y a pas que le secteur du jeu qui a été charmé par les avantages de TUI, il existe par exemple un éditeur de vidéo tangible [29]. Une vidéo est découpée en plusieurs segments et chaque segment est enfermé dans un bloc de séquences vidéos, voir Figure 7. Entre chacun de ces blocs, l'utilisateur va venir accrocher des blocs de transition permettant ainsi de choisir quel type de transition l'utilisateur désire entre ces deux séquences vidéos. Un bouton permet de lancer la vidéo éditée sur un ordinateur et une roulette permet de choisir à partir de quel moment l'utilisateur désire lancer la vidéo.

FIGURE 7 – L'éditeur de vidéos tangibles

Il y a également des applications dans le domaine de la musique. En effet, le monde du tangible possède son propre "instrument" de musique avec la ReacTable[30]. Il s'agit d'une *tabletop* permettant de jouer de la musique en groupe en déplaçant des palais sur la table, voir Figure 8. Le son est modifié par la position des palais, sa rotation, etc... La ReacTable utilise le framework ReacTIVision ainsi que le protocole TUIO présentés précédemment.

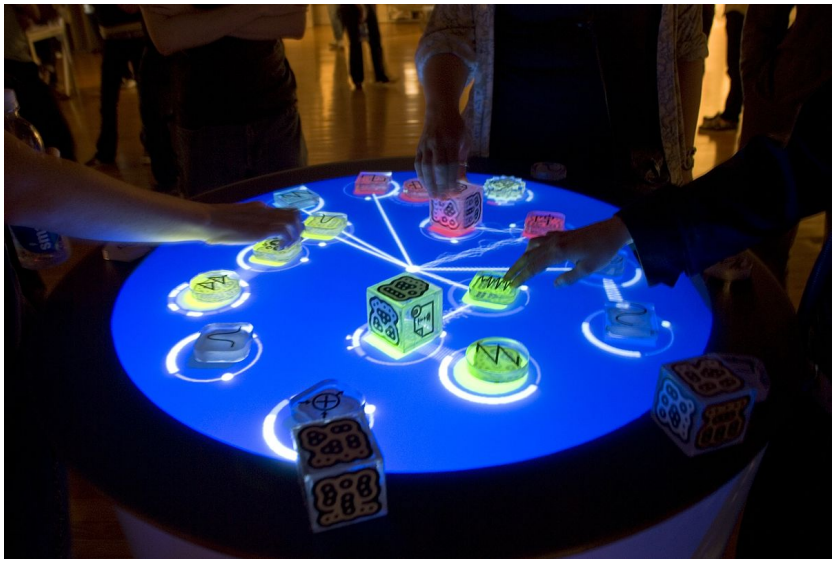


FIGURE 8 – La reacTable

La tangibilité dans les interactions sociales

Les interfaces tangibles sont aussi utilisées dans la communication. Les objets tangibles sont le support idéal pour représenter des personnes. Un exemple de communication est Lumitouch

[31], un cadre connecté qui permet à un couple séparé par la distance de savoir quand son partenaire touche le cadre. Afin de prévenir la personne que l'autre cadre a bien été touché, celui-ci s'allume.

2.2.2 La tangibilité dans l'éducation

Clifford De Raffaele[32] a réalisé une étude sur l'utilisation de la tangibilité dans l'enseignement auprès d'un groupe d'étudiants en informatique en première année apprenant les bases de données. L'outil tangible utilisé ici est une *tabletop* illustrant la normalisation d'une base de données. Les résultats de cette étude ont montré une amélioration des notes des étudiants. De plus, les retours de ces étudiants sur l'expérience ont été plus qu'encourageants. En effet, ceux-ci ont trouvé qu'un cours avec une manipulation d'une interface tangible était bien plus amusant qu'un cours traditionnel. De plus, bénéficier de retours auditifs et visuels en temps réel les aidaient énormément : cela leur permettait de repérer les erreurs directement et donc de les résoudre au plus vite.

Ces résultats sont renforcés par l'étude de Schneider [33] qui obtient des résultats similaires. Il a réalisé son étude auprès d'apprentis du "Centre Professionnel Nord Vaudois". Les apprentis devaient utiliser une interface tangible pour concevoir et designer l'agencement d'un entrepôt. Les résultats de son étude ont montré que l'apprentissage par le tangible présente plusieurs avantages. Le premier est que les actions physiques ont plus de sens, principalement pour les enfants et les novices. L'interaction est plus naturelle et donc est grandement facilitée. Il n'est pas nécessaire d'utiliser régulièrement l'outil tangible pour le prendre en main parfaitement. Ensuite, la correspondance entre les *inputs* tangibles et les *outputs* digitaux permet de réduire l'effort cognitif pour résoudre un problème. Grâce au tangible, il n'y a pas besoin de représenter la réalité à travers différents concepts, la réalité est déjà présente et elle est manipulable, ce qui permet d'enlever un niveau d'abstraction de la réalité et donc de pouvoir se focaliser sur l'essentiel lors d'une résolution de problème. Enfin, les interfaces utilisateurs tangibles supportent et encouragent la collaboration active. Contrairement aux traditionnels claviers/souris qui limitent l'interaction possible à seulement une personne, beaucoup d'interfaces tangibles supportent la collaboration et encouragent la sociabilisation. En effet, lors de l'utilisation d'une telle interface, plusieurs utilisateurs peuvent manipuler différents objets en même temps et donc entrer des *inputs* en même temps, ces interfaces gèrent très aisément des *inputs* simultanés.

Ce dernier point est cependant nuancé par l'étude réalisée par Speelpenning [34]. Selon lui, la collaboration dans l'utilisation d'outil tangible et multi-touches est plus influencée par la dynamique du groupe plutôt que par les outils proposés par l'interface. Pour revenir sur le travail de Schneider, celui-ci présente quelques inconvénients des TUI. D'abord, les manipulations effectuées peuvent être mal interprétées dans le sens où les utilisateurs, et surtout les enfants, ne parviennent pas à comprendre que les manipulations peuvent représenter autre chose. C'est une des raisons pour lesquelles les TUI ne sont pas idéales pour l'apprentissage des mathématiques car les différents symboles peuvent être mal interprétés. Ensuite, l'interface pourrait contraindre la pensée de l'utilisateur. Celui-ci pourrait être bloqué dans un mode d'action, ce qui empêcherait son raisonnement plus abstrait. Il aurait donc plus de mal à prendre du recul sur les tâches qu'il fait. Enfin, avec les TUI, il existe une limite physique. Contrairement à une interface graphique où une infinité d'objets et d'éléments peuvent y être affichés, les interfaces tangibles sont limitées par le nombre d'objets pouvant être manipulés.

Exemples d'outils tangibles dans l'éducation

Il est important de préciser que ces exemples sont principalement utilisés pour l'enseignement de l'informatique, de la programmation ou des mathématiques car il s'agit de domaines très étudiés dans le milieu de la tangibilité. Voici quelques exemples d'applications tangibles utilisées dans le domaine de l'éducation.

Flowblocks [35] est un outil permettant aux enfants d'environ dix ans d'apprendre les bases des concepts de séquences, de boucles, d'embranchements et de statistiques. L'outil est composé

de différents blocs qui peuvent s'imbriquer, voir Figure 9. Il y a un bloc générateur de courant et des blocs pour faire des chemins. La lumière en circulation représente le processus qui circule à travers les différents blocs. Il y a également des blocs de règles qui permettent d'accélérer ou de ralentir le processus. En fonction des embranchements, il est possible que la lumière choisisse aléatoirement son chemin, c'est comme cela que les statistiques interviennent. Après ce test, les créateurs de Flowblocks se sont rendu compte que les enfants comprenaient très vite comment faire des séquences et qu'ils voulaient toujours essayer plus de combinaisons différentes.

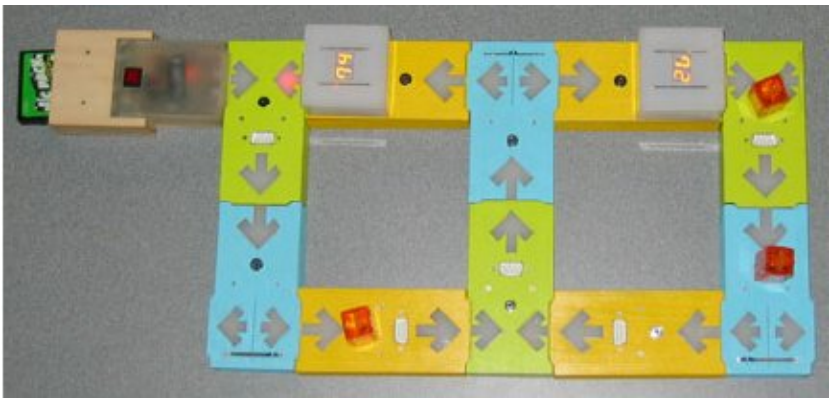


FIGURE 9 – Flowblocks

Tern [36], est similaire à Flowblocks. Il s'agit d'un langage de programmation spécialement conçu pour les interactions tangibles. Il fonctionne avec des blocs en forme de pièces de puzzle afin de pouvoir les emboîter facilement, voir Figure 10. Chaque bloc représente soit une commande, soit une variable. Le fonctionnement des blocs est assez similaire au jeu *Scratch*[5]. Ce langage est conçu comme un outil pour enseigner la programmation auprès des enfants.

FIGURE 10 – Les différents blocs présents dans le langage *Tern*

Il existe une *tabletop* tangible afin de faciliter l'apprentissage de la trigonométrie [37]. L'application consiste en un contrôleur tangible qui représente le cercle trigonométrique qui interagit avec un écran. Le cercle peut glisser sur l'écran, être appuyer contre l'écran, tourner autour de l'écran, voir Figure 11. Cette *tabletop* a été testée auprès d'étudiants en mathématiques avancées. Les participants à l'expérience ont déclaré qu'utiliser plusieurs sens pour découvrir les concepts de la trigonométrie contribuait à diversifier leur accès à l'information. Être capable de toucher

5. https://scratch.mit.edu/

et déplacer la représentation de l'information favorise une meilleure compréhension des concepts parfois très abstraits. L'expérience a également révélé que l'interface tangible influençait positivement leur vision de la trigonométrie, leur permettait de clarifier des représentations erronées qu'ils détenaient et leur permettait d'apprendre en groupe.

FIGURE 11 – TUI pour la trigonométrie

Tanginet [38] est un autre exemple d'interface tangible utilisé en éducation. Il sert à apprendre les différentes propriétés des câbles réseaux. Il y a deux grandes parties dans cette application : une partie exploration, où l'étudiant découvre les différentes propriétés des câbles grâce à un ensemble d'activités, et la partie quizz qui permet d'évaluer le gain de l'étudiant. Trois types d'interactions ont été ajoutés à cette interface tangible. La première consiste à placer différents objets sur un écran (placer un objet représentant une lettre sur l'écran afin de répondre au quizz par exemple). La seconde consiste à déplacer les objets tangibles à un endroit particulier. La troisième interaction consiste à tourner les objets tangibles selon un certain angle. Après les différents tests effectués auprès d'étudiants en réseau informatique, un questionnaire a été réalisé afin de récolter les avis des étudiants. Les développeurs de cette table tangible ont conclu que le système était relativement facile à utiliser. Les participants ont trouvé que le tangible était bien travaillé : ils pouvaient manipuler les objets comme des objets du monde réel et les objets physiques correspondaient à leur représentation digitale.

2.2.3 La tangibilité dans les jeux

Cette section présente différents exemples concrets de jeux tangibles. Il y a deux grandes catégories de jeux : les jeux classiques, du type jeu de plateau par exemple et les jeux utilisant des interfaces utilisateur tangibles. Dans un premier temps, plusieurs exemples de jeux de plateau seront présentés et leurs bénéfices dans l'enseignement seront soulignés. Puis d'autres exemples de jeux utilisant des interfaces utilisateurs seront présentés. Pour chacun de ces exemples, leurs avantages seront également soulignés.

Les jeux de plateau

Tous les jeux de plateau sont par définition tangibles. En effet, ils sont tous constitués de matériels manipulables afin de faire une certaine action, et ainsi faire avancer le jeu. Les objets manipulables dans les jeux de plateau sont variés. Il peut s'agir de cartes, de pions, de dés, du plateau en lui-même etc... Parmi les jeux de plateau les plus populaire, le Monopoly, produit par Hasbro, est un bon exemple pour illustrer la tangibilité. Le but du jeu est d'acheter et d'échanger des propriétés, de les développer en y ajoutant des hôtels ou des maisons. Les joueurs récoltent des loyers des autres joueurs lorsque ceux-ci se trouvent sur leur propriété. Lorsqu'un joueur n'a plus de quoi payer les autres, il est déclaré en banqueroute. Le dernier joueur à ne pas être en

banqueroute gagne[39]. Dans ce jeu, la tangibilité se fait à plusieurs niveaux. D'abord, il y a le pion que le joueur doit manipuler afin d'avancer. Ensuite, il y a les dés, que le joueur doit lancer afin de savoir de combien de case il doit avancer. Puis, il y a l'argent que le joueur doit manipuler afin de gagner des terrains payer les autres joueurs. Enfin, il y a les différentes améliorations de propriétés que le joueur doit ajouter sur ses terrains.

Les jeux utilisant des interfaces utilisateurs tangibles

Les interfaces utilisateurs tangibles sont également un terrain très fertile pour le jeu.

Le premier exemple est *Entertaible*, un jeu qui se veut être l'équivalent électronique des jeux de plateau traditionnels. Il s'agit d'un écran tactile qui permet aux utilisateurs d'interagir simultanément, voir Figure 12. En effet, cette interface est capable de détecter plusieurs entrées simultanées. Le jeu de base pour illustrer les fonctionnalités de cette table est un jeu de taxi. Le but est de contrôler un taxi, représenté par un pion. Le joueur qui a conduit le plus de passagers gagne. Afin de tester la popularité de cette table, les créateurs ont proposé à un groupe d'enfants deux versions du jeu avec les taxis. Une sur la table tangible et l'autre sur un ordinateur avec des graphismes en 3D. Les retours des enfants étaient grandement en faveur de l'entertaible. Ceux-ci la trouvaient plus amusante. Aucune explication des concepts du jeu ou de comment interagir avec la table ont été nécessaires contrairement à son équivalent en jeu-vidéo 3D.[40]

FIGURE 12 – Entertaible

Le second exemple est *ApartGame*. Il s'agit d'une plate-forme de type *Tabletop* qui permet de jouer à différents jeux. Cette plate-forme est composée de cellules qui servent à afficher de l'information, du texte ou une couleur, et qui permettent également aux joueurs d'effectuer des actions en appuyant dessus, voir Figure 13. Un exemple de jeu utilisé par ce support est *Collapse*, un jeu coopératif. Depuis le centre de la table, des blocs de couleurs tombent. Le but est de réunir trois blocs ou plus d'une même couleur afin de créer des combinaisons. Les joueurs doivent appuyer sur un bloc qui tombe afin de changer aléatoirement sa couleur. Dès qu'une combinaison est formée, ils doivent appuyer sur un bloc de cette combinaison pour marquer des points. Plus les joueurs ont de points, plus la vitesse des blocs augmente. Le jeu s'arrête quand plus aucune combinaison n'est possible. L'évaluation de cette table a été effectuée grâce à un questionnaire auprès des participants qui ont utilisé la table lors d'une convention. Il ressort que jouer avec ce jeu est amusant et qu'il est facile à prendre en main. De plus, le jeu a intrigué les joueurs par sa manière originale de jouer et par conséquent les joueurs voulaient toujours tester plus et pousser les limites de l'outil. [41]

Enfin, le dernier exemple présenté dans ce travail est *PlayTogether*, une interface tangible permettant de jouer à des jeux classiques, comme les échecs ou les dames, à distance. Cet outil

FIGURE 13 – ApartGames

projette un plateau d'échec et le premier joueur positionne alors ses pièces physiques, voir Figure 14. Un second joueur va fait ensuite de même. Une caméra filme les pièces des deux joueurs. Lorsqu'un joueur bouge une de ses pièces, le changement est projeté sur le plateau de l'autre joueur. [42].

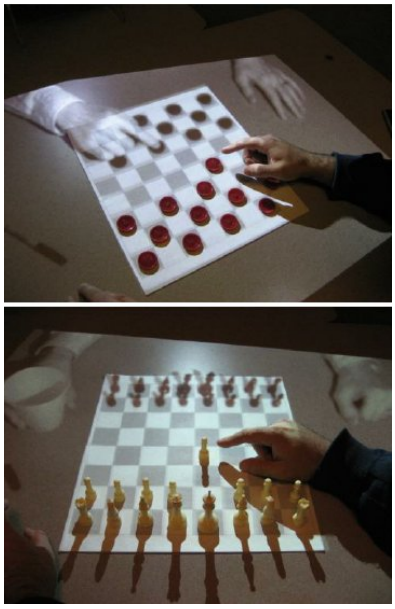


FIGURE 14 – PlayTogether

2.3 L'enseignement de la programmation orienté-objet grâce au tangible

Quand des combinaisons de mots-clés tels que "Learning", "object oriented programming", "tangible" sont recherchés sur Google Scholar, très peu de résultats apparaissent. Beaucoup d'articles traitent plutôt de programmation avec du tangible ou de l'apprentissage de la programmation grâce au tangible. Très peu s'intéressent à l'enseignement de l'orienté-objet par le tangible. Le résultat le plus proche de cette rencontre entre les domaines de l'enseignement de l'orienté-objet et le domaine du tangible est une approche par le jeu de l'apprentissage de l'orienté-objet réalisée par Rodriguez[43]. Son approche consiste en un cours de programmation en C#, un langage de programmation orienté-objet. Dans ce cours, les étudiants apprenant l'orienté-objet sont séparés en deux groupes : ceux qui suivent le cours classique et ceux qui suivent un cours avec des interfaces utilisateurs tangible, les Siftéo Cubes. Il s'agit de cubes tangibles et connectés qui permettent de jouer à une multitude de jeux de manière tangible. Chaque cube est composé d'un écran tactile et chaque face du cube permet, via différents capteurs, de réagir à d'autres faces du cube, voir Figure 15. Lorsque deux cubes sont connectés, ils réagissent ensemble et permettent donc de réaliser différentes actions, comme augmenter l'interface de jeu ou bien encore effectuer une action précise dans le jeu.

FIGURE 15 – Les cubes siftéo

Cette étude met en lumière certains aspects du tangible par rapport à l'orienté-objet. En effet, en plus d'observer une augmentation de la note des étudiants devant programmer sur des interfaces tangibles, ces mêmes étudiants ont montré un intérêt plus grand pour l'orienté-objet par rapport aux étudiants suivant le cours classique.

En conclusion, très peu de recherches traitent de l'utilisation du tangible dans l'enseignement de l'orienté-objet. Par contre, il ressort de la littérature que l'utilisation d'outils tangibles dans l'enseignement présente de nombreux avantages, notamment la collaboration, l'exploration ou plus généralement, une meilleure compréhension de concepts abstraits. De plus, beaucoup d'articles parlent des avantages du tangible dans l'apprentissage de la programmation, ceci montre l'intérêt d'utiliser cette manière d'enseigner en informatique. L'objectif du présent travail de recherche est de s'intéresser au tangible comme support à l'enseignement de l'orienté-objet.

3 Problématique

La thématique de ce travail est "la tangibilité de l'orienté-objet". En d'autres termes, comment utiliser un support tangible afin d'aider à l'enseignement de la programmation orienté-objet. L'idée d'un jeu de rôle est venue très rapidement, mais il fallait encore trouver le support adéquat. Beaucoup de discussions ont été réalisées autour du support à utiliser tels que les cubes *siftéo* ou encore les cartes *micro :bit*. Puis, après discussion avec une autre chercheuse, l'idée d'un jeu de plateau débranché est arrivée.

La problématique de ce mémoire est donc la suivante : "Est-ce que l'utilisation d'un jeu de rôle tangible peut aider à l'enseignement du paradigme orienté-objet ?" L'objectif est de créer un support tangible pour illustrer le fonctionnement de l'orienté-objet. Ce support tangible prendra la forme d'un jeu de rôle plateau.

Ce travail essaiera de vérifier plusieurs hypothèses :

— ce jeu de rôle illustre correctement les concepts orienté-objet ;
— ce jeu de rôle peut s'intégrer facilement à un cours d'orienté-objet ;
— les illustrations des concepts issus du jeu de rôle sont facilement compréhensibles.

4 Méthodologie

Ce chapitre présente la méthodologie utilisée afin de concevoir et de tester ce jeu tangible pour l'apprentissage de l'orienté-objet. Tout d'abord, une introduction théorique sera faite sur le type de méthodologie utilisée. Puis, l'application de ce cadre théorique au jeu tangible sera explicitée.

4.1 Recherche orientée par la conception

La méthodologie de recherche orientée par la conception ou *Design based research* permet de faire de la co-création avec des experts et des praticiens. Il s'agit d'un processus itératif qui articule les phases de conception d'interventions éducatives pouvant prendre la forme d'artefacts, de dispositifs techno-pédagogiques ou de programmes éducatifs, de leur mise en oeuvre à différents niveaux, activité de classe, séance, interventions,...). L'analyse des résultats de ces pratiques éducatives est ensuite réalisée de manière collaborative entre chercheurs et praticiens [44] [45]. Le but de cette méthodologie est d'inclure à la fois les experts et les praticiens lors de la conception et d'analyser le résultat des différentes itérations avec eux afin d'améliorer et de corriger le dispositif pédagogique pour les prochaines itérations.

Différents avantages soutiennent cette approche[45]. D'abord, utiliser le dispositif dans une situation réelle garantit que les résultats peuvent être utilisés efficacement pour évaluer et améliorer la pratique pour les prochaines itérations. De plus, étant donné que les interventions sont testées dans un contexte réel, la mise en pratique réelle est donc grandement facilitée car elle est semblable aux tests déjà effectués. Ensuite, les itérations multiples permettent d'améliorer considérablement le dispositif sur le long terme. Cependant, la difficulté de ce type de design par itération est de savoir quand le programme de recherche est complété.

4.2 Application de la méthodologie

La méthodologie de ce travail s'inscrit dans la recherche orientée par la conception. Elle est synthétisée par le schéma sur la Figure 16.

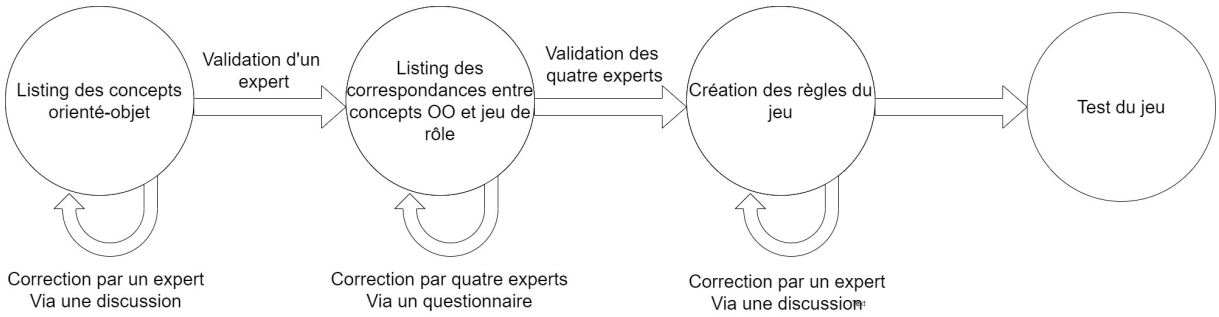


FIGURE 16 – Méthodologie utilisée

Plus précisément, différentes phases balisent notre méthodologie. D'abord, nous avons créé un listing des concepts orienté-objet. Ce listing devait ensuite être validé par un expert. Ensuite, nous avons créé une liste de correspondances entre les concepts de la première phase et des éléments de jeux de rôle. Nous avons ensuite soumis ces différentes correspondances à quatre experts afin de valider cette liste. Puis, nous avons créé les différentes règles du jeu. Ces règles devaient être validées par un expert. Enfin, la dernière étape de ce processus itératif de conception était de tester le jeu.

4.2.1 Premier listing sur les concepts orienté-objets

La première étape était d'effectuer un listing exhaustif des concepts orienté-objets. Nous avons établi un listing sur base du cours Conception et programmation orientée objet [INFOB234] et *Program Development in JAVA : Abstraction, Specification, and Object-Oriented Design* de Barbara Liskov et John Guttag [46]. Il est important de préciser que le travail de Liskov et Guttag se base sur une version de JAVA.

Nous avons parcouru ces sources afin de lister tous les concepts présents. Chaque nom de concept était associé à une courte description. Nous avons ensuite fait lire cette liste par un expert. L'objectif de l'expert était d'apporter ces connaissances afin de compléter cette liste, puis de la valider. Cela fait partie du processus itératif de conception. A chaque retour de l'expert sur la liste, celle-ci est retravaillée jusqu'à la validation de l'expert.

Une fois que la liste de concepts a été définie, il fallait trier ces concepts par ordre d'importance dans l'enseignement de l'orienté-objet. Afin de respecter la méthodologie de co-création avec les experts, deux phases étaient utilisées pour trier les concepts. Lors de la première phase, nous avons défini quatre catégories. Ces catégories sont définies comme ceci :

— la première catégorie comprend tous les concepts sans qui, le paradigme orienté-objet est impossible, ils sont les concepts clés. Par exemple, il est impensable d'imaginer une compréhension de l'orienté-objet sans connaître le concept de classe ou d'objet. Ne pas tenir compte de ces concepts clés provoquerait une compréhension partielle, voire nulle du paradigme ;

— la deuxième catégorie contient des concepts qui se basent sur les concepts de la première catégorie. Il est donc impossible de les comprendre sans comprendre les concepts plus généraux avant. Sans ces concepts, seule une compréhension minimale est possible. Cependant, ces concepts se retrouvent dans la littérature sur les problèmes fréquents des étudiants apprenant l'orienté-objet) comme étant les concepts qui posent le plus de problèmes aux étudiants en cours d'apprentissage de l'orienté-objet (voir sous-sous-section 2.1.3 ;

— la troisième catégorie comprend également des concepts qui se basent sur des concepts plus généraux. Cependant, ceux-ci sont moins liés à des problèmes de compréhension des étudiants. Il s'agit de concepts qui permettent de compléter certains concepts précédents et de pousser leur compréhension ;

— la quatrième catégorie regroupe les concepts qui requièrent déjà une compréhension globale du paradigme. Les concepts plus techniques, comme les concepts en lien avec la machine virtuelle JAVA, sont également présents. En effet, ceux-ci sont propres à l'environnement de développement et sont donc moins pertinents pour l'apprentissage de l'orienté-objet.

Nous avons ensuite réparti les différents concepts parmi ces catégories. Nous avons trié ces concepts en fonction du cours Conception et programmation orientée objet [INFOB234], cours donné aux étudiants de deuxième année en sciences informatiques à l'université de Namur par le Professeur Heymans, mais également en fonction des différents problèmes fréquents rencontrés par les étudiants en apprentissage du paradigme orienté-objet présents dans la littérature.

Ce premier tri était ensuite modifié lors de la seconde phase par un expert. Nous avons entamé une discussion avec un expert afin que celui-ci puisse donner son avis et modifier les catégories des concepts afin que ces catégories soient les plus correctes possibles. Avec les retours de l'expert une nouvelle répartition dans les catégories est alors attribuée. Ce tri était à nouveau donné à l'expert afin que celui-ci effectue une dernière vérification et puisse donner sa validation.

4.2.2 Second listing sur la correspondance entre concept orienté-objet et jeu de rôle

La seconde étape de ce processus itératif de co-création avec experts est de créer une liste reprenant les différents concepts orienté-objet définis dans le premier tableau. Ces différents concepts seront alors associés à des éléments d'un jeu de rôle. L'idée est de "traduire" ces différents concepts afin de déterminer si l'utilisation d'un jeu de rôle tangible pour illustrer les concepts OO est possible.

La création de ce tableau est divisée en quatre phases :

La première phase était la création d'une première version du tableau. Pour pouvoir adapter les concepts orienté-objet vers l'univers des jeux de rôle, il faut d'abord comprendre ce dernier. Les différents éléments appartenant aux jeux de rôle pour faire la correspondance avec les concepts orienté-objet sont issus de plusieurs sources d'inspirations importantes dans l'univers des jeux de rôles. Les inspirations majeures des différents éléments sont assez variées et peuvent prendre différentes formes. Parmi ces inspirations, nous avons utilisé le jeu de plateau "Donjons et Dragons" pour tout ce qui est mécanique de jeu propre à un jeu de rôle plateau. Il y a également la série de jeu vidéo "*The Elder Scrolls*" du studio Bethesda ainsi que la série de roman "Le Seigneur des anneaux" de JRR Tolkien pour la création de l'univers dans lequel les différents concepts évoluent. Cette première version de ce listing se trouve dans l'Appendice B.

Nous avons alors donné ce listing à un expert lors de la seconde phase. Le but était qu'il puisse donner un premier retour sur la correspondance entre concept orienté-objet et élément de jeu de rôle. Une discussion débutait avec lui afin d'en débattre. Sur base de ses retours, une seconde version du tableau sera alors construite pour satisfaire cet expert.

Lors de la troisième phase, nous avons interrogé quatre autres experts afin de s'assurer de la correspondance entre les concepts orienté-objet et le jeu de rôle. Si lors de la phase précédente, l'intervention de l'expert permettait d'enrichir la première version du listing, cette phase consistait à vérifier si la correspondance était évidente. C'est-à-dire est-ce que la correspondance était claire et non-ambiguë. L'expert qui était intervenu à la phase précédente n'a pas été repris dans la présente phase car il avait déjà eu un aperçu de la correspondance proposée dans la première version du tableau et donc sa vision était biaisée. Afin de récolter leur avis, nous leur avions demandé de répondre à un questionnaire. Ce questionnaire a été conçu pour tester la correspondance des concepts. Pour ce faire, deux parties sont mises en confrontation : les concepts orienté-objet du premier tableau et les éléments de jeu de rôle du second tableau. Le but était d'associer les éléments des deux parties et de trouver quel concept orienté-objet correspond le mieux à chaque élément du jeu de rôle. La correspondance établie dans le second tableau était volontairement cachée à l'expert afin que celui-ci puisse associer les différents éléments sans être influencé. L'expert avait également la possibilité de commenter chaque concept s'il trouve que la correspondance n'est pas évidente ou qu'elle pouvait être améliorée. Un exemplaire de ce questionnaire se trouve dans l'Appendice C. L'analyse des résultats de ce questionnaire permettait enfin de confirmer les différentes correspondances ou de les améliorer en fonction des suggestions et remarques des experts.

La dernière phase consistait à corriger le listing en fonction des remarques des experts. Une fois le listing modifié, une dernière validation des quatre experts était nécessaire. Pour ce faire, nous avons envoyé à chaque expert la liste des concepts qui ont été modifiés, avec leur nouvelle description. Après cet échange, les experts devaient valider ou non les modifications. La version définitive de se listing se trouve dans l'Appendice D

4.2.3 Création des règles du jeu

Les règles du jeu se divisaient en deux parties. La première partie était constituée des règles du jeu de rôles, tandis que la seconde partie était composée des différentes règles pour utiliser le jeu comme support lors d'un cours d'orienté-objet. Les deux parties étaient ensuite données à un expert afin d'être validées. Dans notre cas, notre expert était une personne étant déjà familiarisée avec l'univers du jeu de rôle. Cela nous permettait d'assurer à la fois que les concepts

orienté-objet soient correctement représentés tout en ayant un jeu tout à fait jouable et utilisable.

4.2.4 Expérimentation du jeu

Une fois les règles du jeu validées par un expert, la dernière partie de notre méthodologie consistait à tester le jeu. Cette expérimentation s'inscrit toujours dans notre méthodologie de co-création avec des experts. Cependant, au vu du contexte dans lequel ce travail a été effectué (confinement) le test a donc été réalisé en vidéo-conférence via *Teams*. L'expérimentation a été réalisée en deux fois, avec trois participants à chaque fois, afin de récolter l'avis de six personnes au total. Les deux groupes de participants étaient constitués de deux étudiants de la faculté d'informatique de l'UNamur ayant déjà suivi le cours de "Conception et Programmation Orientée Objet" et également un assistant chercheur de la faculté d'informatique. Ce groupe mixte permet à la fois de récolter l'avis d'étudiant sur la mise en place d'un tel dispositif d'enseignement tout en prenant en compte un avis plus pédagogique de la part des assistants. L'expérimentation avait le même déroulé lors des deux représentations afin de réunir les mêmes conditions et le même contexte. Ce déroulé se trouve dans l'Appendice G.

Ce déroulé est composé de deux phases.

La première phase consistait en une partie du jeu de rôle d'environ une heure. Les trois participants incarnaient chacun un personnage tandis que je jouais le rôle de maître du jeu. Lors de cette phase, les règles du jeu étaient rappelées aux joueurs avant le début de la partie. Ensuite, les joueurs évoluaient dans un scénario préparé à l'avance. Ce scénario est une adaptation d'une histoire populaire dans l'univers des jeux de rôles. Il s'agissait de l'histoire "Un chevalier sombre et tempétueux" [6].

La seconde phase consistait en un debriefing de la partie avec les participants. Le but de ce debriefing était de discuter avec les participants des différents concepts orienté-objet présents dans le jeu, mais également de discuter de l'intégration de ce jeu dans le cadre d'un cours d'orienté-objet. Lors de cette phase, deux façons d'utiliser le jeu en support d'un cours étaient présentées. La première façon était de cibler un concept orienté-objet précis et de se référer au tableau de correspondance afin de présenter l'élément de jeu de rôle associé. La seconde façon d'utiliser le jeu était de sélectionner une phase de jeu (la phase de création de personnage par exemple) et de la décortiquer afin de souligner tous les concepts orienté-objet présents. Ces deux façons d'utiliser le jeu sont détaillées dans les règles du jeu, dans l'Appendice E.

Entre ces deux phases, on a demandé aux participants de répondre à un questionnaire d'appariement entre les concepts orienté-objet et le jeu. Dans ce questionnaire, il était demandé aux participants d'associé un concept orienté-objet à un élément le représentant dans le jeu de rôle. Un exemplaire de ce questionnaire se trouve dans l'Appendice H. La liste des concepts et des éléments du jeu de rôle provient du listing corrigé par les experts sur les correspondances entre concepts OO et jeux de rôle. L'objectif de ce questionnaire était de capturer la première intuition des participants. En d'autres termes, l'objectif était de voir si, sans explications supplémentaires, le participant était capable de détecter les concepts orienté-objet présents dans le jeu. Dans l'optique de capter cette première intuition, les participants avaient environ une vingtaine de minutes afin de répondre à ce questionnaire.

6. http ://www.jdrp.fr/telechargement/asmodee/un-chevalier-sombre-et-tempetueux.pdf

5 Développement de la recherche

Cette partie présente la solution proposée pour répondre à la problématique. La problématique de ce travail est : "Est-ce que l'utilisation d'un jeu de rôle tangible peut aider à la compréhension du paradigme orienté-objet ?" .Cette partie reprend notre cheminement de pensée nécessaire qui nous a permis d'arriver à cette solution, ainsi que sa conception.

5.1 Listing des concepts Orienté-Objet importants

Le listing présenté ici est le résultat de la première partie de notre méthodologie. Cet outil contient une liste exhaustive des concepts orienté-objet, ainsi qu'une courte description. Ces concepts ont également été triés en différentes catégories. Ce tableau se trouve dans l'Appendice A. Conformément à la méthodologie définie précédemment, les différents concepts de ce tableau ainsi que leur catégorie ont été validés par un expert dans le domaine.

Afin de comprendre au mieux ce tableau, il est important de détailler les différentes catégories de concepts.

Les concepts de la première catégorie ont été définis comme étant les concepts clés de l'orienté-objet. Sans eux, la compréhension du paradigme orienté-objet est impossible. Dans cette catégorie, les concepts suivants sont présents :

- les concepts de classe et d'objet : il est impossible d'apprendre l'orienté-objet en ne mettant pas ces deux concepts au centre de l'attention. Tous les autres concepts de ce tableau découlent d'eux ;
- les concepts liés aux différentes formes d'abstraction. L'abstraction est un élément au centre de l'orienté-objet. Que ce soit l'abstraction de spécification et par extension les principes de localité et de modifiabilité ou encore l'abstraction procédurale, qui est plus importante dans la programmation au sens large. Cela peut également être l'abstraction de données, qui permet de définir de nouveaux types de données, ce qui est une partie très importante de la programmation en orienté-objet et plus particulièrement en JAVA ;
- le principe de Parnas. Il s'agit d'un principe primordial qui permet de définir correctement une classe ;
- le concept du mutabilité. C'est l'un des concepts les plus important en orienté-objet. Comprendre qu'il existe des objets mutables et d'autres qui sont immutables est essentiel. On ne peut pas échapper à ce concept lors de l'apprentissage de l'orienté-objet ;
- le concept de *Stack et Heap*. Bien que ce concept ne soit qu'un modèle de représentation, il est très utile afin de comprendre correctement le fonctionnement du référencement des objets ;
- les différents concepts liés à l'héritage des classes. Ces concepts sont inévitables lors de l'apprentissage de ce paradigme. Le principe de substitution de Liskov est le principe qui synthétise parfaitement l'héritage. Lorsqu'on étend ce principe, il en découle 3 règles également primordiales par rapport à l'héritage des classes. Il s'agit des règles de signatures, des méthodes et des propriétés.

Les concepts de la seconde catégorie ont été définis comme étant les concepts dont la compréhension requiert la compréhension des concepts de la première catégorie. Ces concepts sont aussi associés à différents problèmes, confirmés par la littérature. Ces problèmes peuvent passer par de l'incompréhension ou par des représentations erronées de la part de ces étudiants. Les concepts de cette catégorie sont les suivants :

- les méthodes. Un manque de compréhension de ce concept chez les apprenants est généralement observés, principalement sur la déclaration de ces méthodes et sur leur invocation. Le rôle de ces méthodes est également mal compris, principalement avec les méthodes de type constructeur. Les méthodes de type observateur, mutateur et producteur sont

également présentes afin d'inclure le plus de notions possibles sur les méthodes et d'éviter au maximum les erreurs de compréhension ;
- le concept de dispatching. Ce concept est lié au concept de méthode. Il s'agit d'un concept important qui permet de comprendre les appels de méthodes avec les méthodes héritées des super-types ;
- les concepts de classes abstraites et classes concrètes. Pour pouvoir les aborder, il faut d'abord introduire le concept de classe avant de présenter les différents types de classes existant ;
- les variables de références. Ce concept est aussi sujet à de l'incompréhension chez les apprenants. Beaucoup ne comprennent pas que deux variables peuvent référencer le même objet par exemple. Les variables primitives ont également une priorité similaire afin de montrer tous types de variables existant et éviter ainsi toutes confusions possibles ;
- les exceptions. Ce concept est aussi régulièrement source de problèmes pour les étudiants. Les comprendre et les définir correctement peut être un vrai casse-tête pour eux.

Enfin, les derniers concepts de cette catégorie sont plus modestes mais sont des extensions de certains concepts de la première catégorie.
- le partage de référence. Même s'il s'agit d'un effet secondaire de la mutabilité, ce concept reste un élément important qui peut poser des problèmes de compréhension s'il est oublié ;
- les types déclarés et les types effectifs. Ces deux concepts sont en lien avec l'héritage et ils sont nécessaires pour comprendre le fonctionnement de ce concept en orienté-objet ;
- les différents concepts dérivés des différentes formes d'abstraction. On retrouve les spécifications, la fonction d'abstraction et l'invariant de représentation. Une spécification est le concept au centre de l'abstraction par spécification. On peut se permettre de le mettre légèrement en retrait car même s'il existe un canevas de spécification en JAVA, on peut quand même les écrire dans un langage informel, qui est suffisant pour comprendre le concept d'abstraction par spécification. La fonction d'abstraction et l'invariant de représentation sont des concepts importants car toutes les implémentations de données abstraites en ont besoin. Il est cependant plus important de comprendre d'abord le concept parent d'ADT (Abstract Data Type), l'abstraction de type de données.

La troisième catégorie reprend des concepts importants mais moins souvent liés à différents problèmes rencontrés par les étudiants. Les concepts suivants sont présents dans cette catégorie :
- le garbage collector. Ce concept permet de comprendre un peu plus en détails le fonctionnement de la *Stack et Heap* et de comprendre ce qui se passe lorsqu'un objet n'est plus référencé ;
- l'encapsulation. Bien que ce concept soit très présent dans la programmation orienté-objet, ce concept peut passer légèrement au second plan afin de privilégier d'autres concepts avant de commencer celui-là. Pour comprendre la visibilité des classes et des méthodes, il faut d'abord comprendre à quoi cela correspond ;
- le casting. L'explication pour ce concept est similaire à celle du point précédent. Pour bien comprendre le casting, il faut d'abord comprendre les types effectifs et types déclarés. Globalement, il faut d'abord comprendre l'héritage ;
- l'autoboxing et l'unboxing. Ces deux concepts ont besoin d'une compréhension des différents types primitifs de JAVA et plus particulièrement des types d'objets primitifs ;
- l'Overloading. Plusieurs points évoqués précédemment sont nécessaires pour aborder correctement l'Overloading, des points comme les méthodes, les types primitifs et les types d'objets ;
- l'adéquation de type. Pour avoir un type adéquat, il faut qu'il possède au moins 3 des 4 catégories de méthodes (créateur, producteur, observateur, mutateur). Il faut donc voir d'abord ces différents types de méthodes avant tout ;
- les méthodes equals, toString et clone. Ces méthodes indiquent le fait que tous les types d'objets sont des sous-types du super-type *Object*. Cela a donc très peu de sens d'introduire ces notions sans parler d'héritage au préalable.

Enfin, voici les concepts évalués avec le degré d'importance le plus faible. Cela ne veut pas dire que ces concepts ne sont pas importants à la programmation orienté-objet, mais plutôt qu'ils ne sont pas les concepts à enseigner en priorité car ils se basent pour la plupart sur d'autres concepts. Il faut donc enseigner d'abord les concepts "pré-requis" avant d'aller plus dans le détail.

— le Type checking et le Type Safety. Ces mécanismes sont finalement plus liés au compilateur plutôt qu'à la programmation orienté-objet. Pareil pour l'Automatic Storage Management qui est un des mécanismes assurant le Type safety, qui lui s'occupe d'allouer la mémoire dans le heap et de gérer le garbage collector ;
— l'abstraction de l'itération. Ce concept requiert plusieurs autres concepts pour être compris correctement, notamment les méthodes, les procédures et les interfaces ;
— les interfaces. Les interfaces sont des types de classes particulières. Il faut donc d'abord se concentrer sur le concept de classe.

Ce listing a été validé par un expert. L'expert sollicité a cependant apporté certaines modifications. Certains concepts ont changé de catégorie afin que ce listing colle le plus possible à la réalité. Par exemple, les types primitifs et types d'objets primitifs étaient initialement présents dans la troisième catégorie. Cependant, après une discussion avec l'expert, ils ont été requalifiés en tant que concepts de seconde catégorie. En effet, ces concepts sont fortement liés au langage de programmation JAVA. Notre solution devant s'intégrer à un cours d'orienté-objet qui utilise JAVA comme langage, il est important de parler de ces deux concepts.

5.2 Second listing sur les correspondances entre concepts orienté-objet et jeu de rôle

Première itération du listing

Cette section présente la première itération du tableau de correspondance entre concepts orienté-objet et jeu de rôle. Ce tableau est issu du résultat de la seconde phase, suite à la discussion avec un expert. Ce tableau vise à démontrer la possibilité d'utiliser un jeu de rôle tangible dans l'apprentissage des concepts clés de l'orienté-objet. Si par exemple, il est impossible de trouver une correspondance entre un concept et le jeu, il faudra alors trouver une autre approche que celle du jeu de rôle. Ce tableau sera ensuite soumis à quatre autres experts afin d'être validé.

Voici la première itération de ce listing de correspondances entre concepts orienté-objet et éléments de jeu de rôle :

Concept	**Description**
Classe	Les classes représentent les personnages et leur fonction. Il y a aussi des classes qui représentent les armes. Une super classe Personnage possède des sous classes Chevalier/Mage/Archer par exemple
Classe concrète	Les classes Archer, Mage et Chevalier sont des classes concrètes, elles peuvent être instanciées
Classe abstraite	La classe Personnage est une classe abstraite, il ne peut pas y avoir un joueur dont le rôle est "Personnage"
Objet	Les objets représentent les différents acteurs/personnages de la partie, mais également leurs armes. Par exemple, il y a Merlin le mage, Robin l'archer et Arthur le chevalier. Arthur est équipé d'Excalibur, son épée
Encapsulation	Le principe d'encapsulation est respecté mais implicite. Les différentes méthodes des personnages ont des visibilités
Principe de Parnas	Traduire le principe en règle du jeu : "La définition d'un rôle doit fournir toutes l'informations nécessaires pour manipuler correctement un personnage de ce rôle"
Type primitif	Les variables d'instance (nom, point de vie, point d'attaque, etc...) sont des types primitifs
Type d'objet	Les personnages et les rôles sont des types d'objets. L'arme du personnage est un type objet également
Variables primitives	Attributs du personnage (Force, dextérité, charisme, nom,...)
Variables de référence	Représentent la vie des personnages. Lorsqu'ils meurent, ils perdent leur référencement
Stack et Heap	Tableau récapitulatif de l'état des joueurs
Partage de référence	Le coffre/inventaire partagé entre les différents personnages
Garbage collector	Le garbage collector est comme la faucheuse, qui vient chercher les personnages morts (plus référencés) pour les emmener dans l'au-delà
Mutabilité	Les personnages sont mutables (leur état peuvent changer, ils peuvent perdre de la vie, gagner en force,...) tandis que les armes sont immutables
Méthodes	Les méthodes sont les actions pouvant être effectuées par les personnages
Type Checking	Présent dans les règles du jeu, un archer n'a pas le droit d'avoir une épée en tant qu'arme principale par exemple
Type Safety	Non présent
Automatic storage management	Non présent
Héritage	Personnage est le super type, les différents rôles sont les sous types. Dans les règles du jeu, le principe de substitution doit bien être respecté, ainsi que les différentes règles (signature, méthode, propriété) également
Principe de substitution (Liskov)	Traduire le principe en règle du jeu : Par exemple : Si le grand mage est un sous type de mage (un mage particulier avec plus de pouvoir) alors le grand mage doit pouvoir effectuer toutes les actions qui requiert un mage
Type déclaré et Type effectif	Ces concepts dépendent fortement du langage utilisé. Ils ne seront donc pas présents dans la première version du jeu

Casting	Des items permettent de faire du casting. Par exemple, un gant spécial permettant de tenir un bâton, un arc ou une épée et donc de donner à une épée le pouvoir d'un bâton de mage
Règle des signatures	Respectée dans les règles du jeu
Règle des méthodes	Respectée dans les règles du jeu : par exemple une méthode "marcher" de Personnage aura le même fonctionnement que la méthode "marcher" d'un chevalier
Règle des propriétés	Respectée dans les règles du jeu : par exemple une méthode "attaquer" de Personnage qui fonctionnerait que lorsqu'il y aurait un ennemi, la même méthode dans Chevalier ne fonctionnera que dans les mêmes conditions
Dispatching	Non présent
Overloading	Non présent
Type d'objet primitif	Non présent
Autoboxing/Unboxing	Non présent
Abstraction par spécification	Respectée dans les règles du jeu, l'implémentation des méthodes est cachée, le joueur doit se repérer avec les spécifications
Spécification	Décrit le comportement des actions des personnages ainsi que le contexte d'utilisation de ces méthodes, écrit en français
Principe de localité	Non présent
Principe de Modifiabilité	Non présent
Abstraction procédurale	Des méthodes seront des procédures, comme la méthodes pour avoir les règles du jeu par exemple
Exceptions	Si le joueur utilise une méthode sans y être autorisé pour une raison ou une autre, une exception est levée et annule son action/le pénalise
Abstraction de données	Un type de données FichePersonnage peut être créé assez facilement. Il représente toutes les informations qu'on a sur un personnage (comme dans un jeu de type donjons et dragons). Chaque personnage aurait un attribut qui serait de type FichePersonnage. Fiche de personnage est immutable, on le définit au début puis il reste la même jusqu'au bout.
Méthode equals, clone et toString	Non présent
Fonction d'abstraction	Non présent
Invariant de représentation	Non présent
Créateur/Constructeur	Lors de la conception du personnage, un constructeur est utilisé
Producteurs	Pour les armes (type immutable), il peut exister une méthode pour dupliquer une arme
Mutateurs	Une méthode comme "soigner" permet de modifier l'état du personnage
Observateurs	Une méthode comme getFichePersonnage() qui permet d'afficher la fiche du personnage

Adéquation de type	Présent dans les règles : l'adéquation de type nous assure qu'un personnage qui est créé est jouable (il peut attaquer, parler, interagir,...)
Abstraction de l'itération	Non présent
Interface	Non présent

Plusieurs concepts de ce tableau méritent des compléments d'informations.

Comme vu dans la sous-section 5.1, les deux concepts les plus importants sont les concepts de classe et d'objet.

Pour les classes, l'idée est la suivante : tout l'univers dans lequel évolue les joueurs est représenté par un ensemble de classes. Les classes représentent entre autres les différents rôles de héros que les joueurs peuvent incarner et les différents types d'armes que les joueurs peuvent utiliser. Par exemple, il y a une classe *Héros* qui est une classe abstraite qui possède des méthodes et des attributs. Cette classe est un super type de plusieurs classes concrètes, par exemple les classes Mage, Chevalier ou encore Archer. Dans l'idée, ces classes sont mutables car l'état du joueur peut et doit évoluer au cours de la partie, afin de montrer une progression. Il y a également des classes non-mutables, par exemple, la classe Arme, qui étendrait des classes Bâton, Arc ou encore Épée par exemple. En effet, les armes resteront identiques et auront le même état tout au long de la partie.

Un objet est le personnage réel incarné par le joueur. Afin de rendre tangible la notion d'objet, différentes figurines seront utilisées. Des figurines représenteront tous les éléments réels de l'univers. La métaphore entre instance (objet) et figurine est assez claire car les objets sont manipulables comme une figurine, là où une classe n'est pas manipulable.

Enfin, certains concepts du tableau n'ont pas d'équivalent dans le monde du jeu de rôle. Voici une explication pour chacun de ces concepts non-présents.

Le *type safety* est un mécanisme de JAVA intervenant à la compilation permettant de vérifier qu'il n'y a aucune erreur de typage. Le traduire en langage de jeu est très compliqué tant ce concept est propre à l'orienté-objet et plus particulièrement aux langages similaires à JAVA. Ce n'est donc pas très pertinent d'intégrer cela, surtout qu'il ne s'agit pas spécialement d'un concept prioritaire dans l'apprentissage de l'orienté-objet.

L'*automatic storage management* est un des mécanismes permettant d'assurer le *type safety* et est géré par la JVM. Pour les mêmes raisons que le point précédent, cela n'a pas beaucoup de sens de l'intégrer au jeu.

Pour ce qui est du *dispatching*, *overloading*, *type d'objet primitif* et *l'autoboxing/l'unboxing*, l'explication est la même, aucune métaphore pertinente n'a été trouvée pour ces concepts. Cependant, même si ces concepts sont importants dans l'orienté-objet, ces thèmes sont rarement à l'origine des problèmes chez les apprenants.

Le *principe de localité* et le *principe de modifiabilité* ont tous deux le même argument. Dans notre jeu, l'étudiant n'est pas voué à manipuler ou écrire du code, mais à manipuler des concepts. Ces deux concepts sont à prendre en compte à un moment où on écrit du code, ils ne sont donc pas très pertinents à intégrer à notre jeu.

L'explication est plus ou moins similaire pour *les méthodes equals, toString et clone*, la *fonction d'abstraction* et l'*invariant de représentation*, qui sont des concepts très liés à la programmation en elle-même. En effet, on estime que le tangible n'est pas très pertinent pour l'apprentissage de ces concepts, afin d'apprendre correctement ces concepts, il faut pratiquer et pour pratiquer, il faut écrire du code, définir des nouveaux types, etc...

L'*abstraction de l'itération* n'est pas présente dans notre tableau. L'explication est la suivante, ce concept apparaît très tard dans l'apprentissage de l'orienté-objet. Donc, pour ne pas compliquer l'apprentissage des concepts plus "élémentaires", on préfère ne pas intégrer l'abstraction de l'itération tout de suite. On peut cependant la garder de côté pour une éventuelle version deux de notre application. On peut par exemple l'intégrer dans un parcours filtré de l'inventaire.

Avis des experts

L'analyse des réponses des questionnaires a été faite en plusieurs étapes. Tout d'abord, une analyse globale du questionnaire a été effectuée. Le taux de correspondance globale ainsi que différentes remarques des experts sont présentés. Puis, les taux de correspondance par catégo-

rie de concepts orienté-objet sont également présentés. Enfin, une conclusion de cette analyse est présentée. Sur base de cette analyse, deux options peuvent être faites. Soit le questionnaire valide le tableau (quelques modifications sont possibles en fonction des recommandations des experts), soit le questionnaire ne valide pas le tableau, et dans ce cas, un nouveau questionnaire doit être effectué avec de nouveaux experts.

Le taux de correspondance globale du questionnaire est de 63%. Ce score a été calculé comme ceci : chaque questionnaire s'est vu attribuer une note (un point par concept correctement associé à sa description). Puis, on a calculé la moyenne de tous les questionnaires.

Deux facteurs expliquent ce taux. Le premier est la difficulté du questionnaire. Il s'agit de la principale critique des experts. Il est difficile d'apparier deux éléments quand tant éléments différents sont présents. Le second facteur est la similitude de certains concepts. La différence entre certains concepts se trouve dans des détails. Par exemple, les experts ont eu tendance à proposer le concept "Abstraction par spécification" à la place de "Spécification". Cette confusion entre ces deux concepts s'explique par la similitude entre les éléments du jeu de rôles de ces deux concepts.

Les concepts de catégorie 1 ont un taux de correspondance de 64%. Une analyse plus détaillée de ces concepts montre que seulement quelques concepts posent des problèmes d'appariement. Il faut donc modifier la description dans l'univers du jeu de rôle de ces concepts afin d'avoir un meilleur rendement. Les concepts posant problème sont énumérés ci-dessous. Pour chacun de ces concepts, une description du problème ainsi qu'une solution à ce problème sont dressées :

- le type d'objet : la définition en jeu de rôle de ce concept est "Définit l'ensemble des éléments du même genre. Par exemple, l'ensemble Arthur le chevalier, Merlin le mage et Robin le chevalier sont tous du genre Héros". Le taux d'appariement de ce concept est de 25% ; La principale critique des experts est que plusieurs concepts sont présents dans cette définition, cela devient donc ambigu pour les experts de devoir cibler un seul concept. Afin de résoudre ce problème, il faut revenir à la définition d'un type objet. Cette dernière se base sur l'aspect ensemble d'un objet, un type d'objet est une collection d'objets, un type d'objet contient l'ensemble des objets d'un même type. La définition devient donc "Définit les ensembles d'éléments du monde réel du même genre. L'ensemble Héros contient tous les éléments réels définis comme des Héros : Arthur le chevalier, Merlin le mage ou Robin l'archer. D'autres ensembles existent également, comme l'ensemble des items existants dans le monde. Ces ensembles peuvent contenir des sous-ensembles, comme l'ensemble des armes qui est un sous-ensemble de l'ensemble des items" ;
- l'abstraction par spécification : la définition en jeu de rôle de ce concept est "Le fonctionnement des différentes actions des héros est caché. Le joueur n'a accès qu'à leur définition". La critique de cette définition est qu'elle est trop proche de la définition de l'encapsulation. Pour remédier à ce problème, il faut recentrer la définition sur les spécifications, afin qu'elle ne soit pas trop vague et trop large. Cette définition devient donc "La description des différentes actions des personnages est suffisante pour comprendre leur fonctionnement et leur résultat." ;
- la règle des propriétés et la règle des méthodes : la définition en jeu de rôle du concept de règle des propriétés est "Une action de Héros "attaquer" qui ne fonctionnerait que lorsqu'il y a un ennemi devant fonctionnera sous les mêmes conditions si c'est un chevalier qui l'utilise" tandis que la définition de la règle des méthodes est "Si des actions sont communes à tous les héros, alors elles auront le même fonctionnement, peu importe le rôle du héros qui l'utilise". Ces deux définitions sont mises ensemble dans l'analyse car les experts ont eu tendance à confondre ces deux concepts. En effet, ces deux définitions sont assez proches sémantiquement, c'est donc logique qu'il peut y avoir une confusion entre les deux. Dans le cadre du jeu, ce n'est pas réellement un problème s'il peut y avoir une confusion entre ces deux concepts. En effet, il s'agit de concepts très peu tangibles, ils ne sont donc pas au coeur du travail. Cependant, ils devront être respectés dans les règles du jeu ;

— l'abstraction de données : La définition en jeu de rôle de ce concept est "Chaque Héros possède des caractéristiques (attaque, défense, vitesse). Ses caractéristiques sont écrites sur une fiche de personnage. Chaque héros possède une fiche de personnage unique". Dans ce cas, le problème est la confusion avec le concept d'objet. Il s'agit d'une confusion logique car un élément appartenant à un type de donnée abstrait est également un objet. Afin de pallier à ce problème, il faut davantage centrer cet élément sur la définition du concept orienté-objet, et donc sur la notion de donnée. Il faut bien montrer que ces types de données doivent représenter de l'information. Cette définition devient donc "Deux types d'informations appartiennent aux joueurs. Le premier type est une fiche de personnage, il s'agit de l'ensemble des informations relatives à un héros, cette fiche contient le nom du personnage, ses points d'attaque, ses points de défenses et ses points de vitesse. Le deuxième type d'informations appartenant aux héros est son inventaire. Un inventaire est une collection d'items ramassés par les héros".

Les concepts de la deuxième catégorie ont un taux d'appariement de 62,5%. Encore une fois, ce taux s'explique par quelques concepts problématiques. Une nouvelle définition de ces concepts est donc nécessaire. Les concepts posant problème sont énumérés ci-dessous. Pour chacun de ces concepts, une description du problème ainsi qu'une solution à ce problème sont dressées.

— le type déclaré : la définition en jeu de rôle de ce concept est "Au début de la partie, avant même que le joueur choisisse son rôle, son personnage existe mais il fera partie de la catégorie Héros". Cette définition est un peu floue et les quatre experts interrogés ne savaient pas avec quel concept l'associer. Par conséquent, nous avons décidé de retirer ce concept du jeu car il est très difficile de le représenter par des éléments de jeu de rôle ;
— le type effectif : la définition en jeu de rôle de ce concept est "Une fois que le joueur a choisi son rôle, le jeu redéfinit son personnage en fonction de ce rôle". Encore une fois, cette définition est floue pour les experts. Ils ont eu tendance à confondre cette définition avec le concept de casting. Nous avons également décidé de retirer ce concept du jeu pour les mêmes raisons que le type déclaré ;
— le partage de référence : la définition en jeu de rôle de ce concept est "Les héros possèdent un inventaire commun dans lequel ils pourront stocker les objets de quêtes ou des armes". Ce concept souffre d'une confusion avec le concept de mutabilité. En effet, ces deux concepts sont liés et une confusion est donc logique. Afin d'y remédier, une nouvelle définition est nécessaire. Cette nouvelle définition doit être plus centrée sur le coeur de ce concept, à savoir deux variables qui référencent le même objet, ce qui peut entraîner des modifications. La nouvelle définition est donc "Tous les héros possèdent un inventaire en commun dans lequel ils pourront stocker les objets de quête ou des armes qu'ils ramasseront au cours de la partie. Cet inventaire est le même pour tous les héros, par conséquent, si un héros retire un objet de l'inventaire, il sera effectivement en dehors de l'inventaire pour tous les héros".
;

Les quelques concepts de la troisième et quatrième catégorie présents dans ce listing présentent un taux d'appariement de 85%. Seul un concept pose problème, il s'agit de l'adéquation de type. La définition en jeu de rôle de ce concept est "Permet de s'assurer que tout héros créé est jouable. Il peut parler, attaquer, interagir avec son environnement, ..". La principale critique de cette définition est qu'elle était trop courte pour déterminer correctement le concept associé. Il faut donc l'expliciter d'avantage. Cette définition devient donc "Permet de s'assurer que tout héros créé est utilisable correctement. Un héros est utilisable s'il fournit un ensemble d'actions permettant aux joueurs de réaliser tout ce qu'il désire avec les héros de manière raisonnable et efficace".

Au vu des résultats, réaliser un nouveau questionnaire avec quatre nouveaux experts n'est pas nécessaire. Cependant, il faut que les quatre experts approuvent les différentes modifications présentées ci-dessus. Afin que ces modifications soit validées nous avons envoyées aux quatre experts un tableau reprenant tous les concepts qui posaient problèmes. Ce tableau contient l'ancienne

description et la nouvelle description. Ils pouvaient également définir une nouvelle description si la nouvelle ne leur convenait pas. Le listing des différentes correspondances corrigées et validées par les experts se trouve dans l'Appendice D.

5.3 Solution proposée

Cette section présente le jeu de rôle tangible qui répond aux besoins de la problématique. Le jeu est présenté grâce à une description, son but et ses règles.

5.3.1 Description du jeu

Notre jeu est un jeu de rôle destiné à servir de support à l'enseignement de l'orienté-objet. Ce jeu de rôle est à intégrer dans un cours d'orienté-objet et est destiné à des apprenants en CPOO. Ce jeu servira pour illustrer les différents concepts orienté-objet lors d'un cours CPOO.

Un jeu de rôle se définit comme une activité par laquelle une personne interprète le rôle du personnage imaginaire dans un environnement fictif. Le joueur doit interpréter le rôle de son personnage. L'interprétation du rôle implique que le joueur doit penser et agir comme son personnage. Dans ce jeu de rôle, les joueurs incarnent des Héros et ils doivent s'unir pour résoudre une quête. L'aventure est racontée par un meneur de jeu (MJ). Le MJ est responsable de la fiction et supervise le jeu. Le MJ doit adapter l'aventure en fonction des choix des joueurs. Les joueurs et le MJ doivent connaître les règles du jeu de rôle avant de commencer la partie.

Le côté tangible du jeu sera apporté par un plateau de jeu qui représentera le monde dans lequel évolue les joueurs, mais également grâce à des figurines représentant les personnages des joueurs. Les joueurs devront également utiliser des dés pour pouvoir effectuer tout un tas d'action.

5.3.2 But du jeu

Le but est d'aider les enseignants et étudiant à mieux appréhender l'apprentissage CPOO via le jeu et une illustration de ce dernier. Les apprenants apprendront à faire le lien entre le jeu de rôle et des concepts orienté-objet. En effet, le jeu a été conçu pour être cohérent avec les différents concepts qui régissent l'orienté-objet. Utiliser des exemples issus du jeu permet aux étudiants de comprendre plus facilement ces exemples, car ils sont familiarisés avec l'univers du jeu, et permet de rendre les concepts concrets et réels. L'analogie permet de mieux comprendre et vérifier la compréhension des étudiants lors d'un cours de CPOO.

5.3.3 Règles du jeu

Le document complet contenant les règles du jeu se trouve dans Appendice E. Ce document contient trois éléments :

— des règles et des conseils pour utiliser le jeu dans le cadre d'un cours d'orienté-objet ;
— le listing des correspondances entre orienté-objet et leur équivalent en élément de jeu de rôle, il s'agit du tableau issu de la deuxième étape de notre méthodologie. Nous ne reviendrons pas sur ce listing dans cette section ;
— les règles d'un jeu de rôle prêt à l'emploi.

Règles et conseils

La première partie sur les règles et les conseils a été conçue comme un mode d'emploi. Le but est de pouvoir fournir uniquement ce document aux enseignants pour qu'ils puissent utiliser directement ce jeu dans leur cours. Ce document détaille comment utiliser le tableau de correspondance afin de faire les liens entre le jeu et les concepts orienté-objet, comme indiqué sur la Figure 17. Ce tableau associe un concept orienté-objet avec un élément de jeu de rôle qui le représente. Ce tableau contient les concepts principaux de l'orienté-objet(1). Il associe

les concepts à un élément d'un jeu de rôle. Puis, il met en pratique les correspondances dans le jeu(2). Ce document met également en lumière les deux façons de se servir des différentes

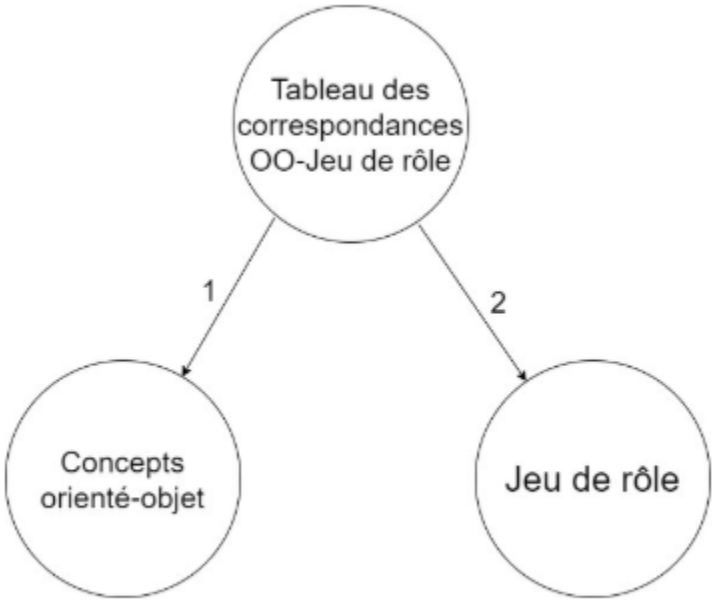


FIGURE 17 – Lien entre jeu et concepts OO

correspondances afin d'illustrer les concepts orienté-objet lors d'un cours. Soit l'enseignant veut illustrer un concept particulier, dans ce cas il doit chercher ce concept dans le tableau des correspondances et ensuite aller chercher la mise en pratique de ce concept dans le jeu. Soit l'enseignant prend un élément du jeu, et le déconstruit grâce au tableau des correspondances afin de montrer à ses étudiants tous les concepts représentés dans cette partie du jeu. Ces deux façons sont également accompagnées d'un exemple d'illustration.

Règles jeu de rôle

Afin de concevoir la seconde partie, nous avons utilisé le système de jeu de rôle BaSIC [7]. Le système BaSIC est un système de jeu de rôle datant des années 70 publié par l'éditeur américain Chaosium. Nous avons choisi ce système car il est réputé pour sa simplicité. La simplicité du système était un critère important lorsque nous devions choisir un système car les règles devaient être facilement compréhensibles, même par les joueurs qui expérimentaient un jeu de rôle pour la première fois. Nous avons adapté ces règles en fonction du listing de correspondances présentés précédemment. Certaines des règles de ce système ont également été simplifiées afin de coller le plus possible à nos besoins.

5.4 Résultat de l'expérimentation

L'expérimentation avait pour but de tester le jeu. Le déroulé de cette expérimentation est détaillé dans l'Appendice G. Lors de cette expérimentation, il était demandé aux participants de répondre à un questionnaire d'appariement, un exemplaire de ce questionnaire se trouve

7. https://www.sden.org/IMG/pdf/basicregles.pdf

dans l'Appendice H. L'objectif de ce questionnaire est de capturer la première intuition des participants sur les concepts orienté-objet et leur représentation dans le jeu.

Les résultats du test d'appariement montrent un taux de 66% de correspondance. Ce résultat a été calculé en faisant la moyenne des concepts correctement associés à leur représentation dans le jeu de chaque questionnaire. Ce résultat signifie que 66% des concepts identifiables dans le jeu, et ce sans aucune explication supplémentaire. Le Tableau 4 indique le taux de correspondance par concept :

Table 4 – Résultat détaillé de l'expérimentation

Concept orienté-objet	**Taux de correspondance**
Classe	83%
Classe concrète	100%
Classe abstraite	50%
Objet	83%
Encapsulation	83%
Principe de Parnas	67%
Type primitif	100%
Type d'objet	17%
Variable primitive	83%
Variable de référence	17%
Stack et Heap	50%
Partage de référence	50%
Garbage Collector	100%
Mutabilité	100%
Méthodes	83%
Type Checking	67%
Héritage	67%
Principe de substitution (Liskov)	67%
Abstraction par spécification	83%
Spécifications	83%
Abstraction procédurale	0%
Exception	83%
Abstraction de données	50%
Constructeur	100%
Producteur	33%
Mutateur	100%
Observateur	33%
Adéquation de type	17%
Total	**66**%

On remarque que la grande majorité des concepts a été correctement associée. Les concepts qui n'ont pas pu être identifiés sont les concepts de classe abstraite, de type objet, de variable de référence, d'abstraction procédurale, d'abstraction de données, de producteur, d'observateur et d'adéquation de type.

Les concepts qui n'ont pas été clairement identifiés le sont parce que ces concepts sont difficiles à être représentés par le jeu de rôle. Par conséquent, il est plus difficile pour les joueurs de les identifier. Même si des correspondances existent, elles ne sont pas aisément perceptibles par les joueurs. L'enseignant utilisant le jeu dans son cours doit donc porter attention à ces concepts lors de la présentation d'exemple issus du jeu. Une solution possible serait de travailler sur un scénario de jeu de rôle mettant en avant ces concepts.

Enfin, l'expérimentation se terminait par une discussion afin de récupérer les avis et les remarques des différents participants. Il s'agit ici donc de remarques plus subjectives et qualitatives. Dans un premier temps, ce jeu a été qualifié d'amusant et de ludique. Les participants se sont bien pris au jeu et ils ont passé un moment amusant.

"Le jeu est vraiment amusant, c'est une manière originale de découvrir l'orienté-objet. Une fois qu'on s'est bien intégré au jeu, on commence à se libérer et tenter des actions de plus en plus folles. J'ai vraiment passé un bon moment, c'était ludique et agréable."

Ce côté ludique est important à prendre en compte lorsqu'on utilise des jeux à but éducatif, car il s'agit d'un facteur pouvant motiver les étudiants lors de l'apprentissage.

Un autre point lors de cette phase de discussion était qu'avoir un exemple récurrent tout au long des séances de cours et des travaux pratiques était une bonne idée.

"Je pense que le gros point fort de ce jeu est de réunir tous les exemples du cours d'orienté-objet en un seul gros exemple. Lorsqu'on suivait le cours, on utilisait beaucoup d'exemples différents comme l'exemple d'Amazon qui était assez récurrent, l'exemple des animaux ou encore les ensembles avec les Multiset, etc. Certains exemples demandaient déjà un peu de temps à être correctement compris car ils étaient très abstraits. Ici comme tous les exemples sont issus du même univers, cela permet une continuité et de ne pas devoir réapprendre un nouvel exemple à chaque fois. Les exemples sont aussi très concrets car on a joué juste avant au jeu et donc on sait ce que représente l'exemple. On peut même imaginer devoir implémenter les différentes classes issues du jeu lors d'un TP par exemple pour rendre l'exemple encore plus concret."

Lors du cours d'orienté-objet, beaucoup d'exemples très différents sont utilisés et il se peut que les étudiants prennent du temps afin d'assimiler correctement ces exemples. Avoir un exemple qui serait déjà compris par les étudiants, car ils auraient joué au jeu avant permettrait de gagner du temps sur l'introduction d'exemple. Utiliser des exemples issus du même univers tout au long des cours théoriques est un plus, car cela permet de créer de la cohérence et de la continuité. Cela permet également d'ajouter du concret aux concepts.

Pour terminer, malgré le fait qu'une partie soit cadrée par un scénario et une histoire, les joueurs sont libres de faire ce qu'ils veulent.

"Ce qui est intéressant c'est que le jeu de rôle donne énormément de liberté. Et quand un étudiant fait une action qui sort un peu de l'ordinaire, qui sort un peu du scénario, ça peut être bien après coup de revenir sur cette action. L'idée est de leur demander de revenir sur cette action là et de dire, au niveau de l'orienté-objet, les concepts qui sont derrière. Comme ça l'étudiant part de son action et on l'invite à réfléchir sur l'orienté-objet en ajoutant une dimension personnelle."

Dès lors, on peut utiliser cette liberté pour créer des exemples personnalisés pour les joueurs. Par exemple, si un joueur fait une action particulière qui sort un peu du cadre de l'histoire racontée par le MJ, on peut lui demander de revenir sur son action et de la réfléchir d'un point de vue orienté-objet en pointant les différents concepts qui ressortent de son action.

Cette expérimentation permet de dégager les grandes tendances par rapport au jeu. Utiliser ce jeu dans l'enseignement de l'orienté-objet offre plusieurs avantages. Il s'agit d'une manière amusante et originale d'aborder l'orienté-objet. Il permet de créer de la cohérence dans les exemples utilisés lors d'un cours théorique. Enfin, le jeu offre suffisamment de liberté pour créer des exemples propres à chaque étudiants.

Cependant, il est évident qu'un test en condition réelle est nécessaire. Dans les tests réalisés ici, les participants étaient des étudiants en deuxième année de master en informatique et des chercheurs en informatique. Autrement dit, il n'y avait aucun novice en orienté-objet. De plus, le test d'appariement ne permet pas de dire si les participants ont eu une meilleure compréhension des concepts grâce au jeu. Il est donc primordial de réaliser un test en classe avec des apprenants en orienté-objet. Afin d'évaluer et de quantifier le gain de compréhension des concepts grâce au jeu, il faut que les étudiants répondent à un pré-test portant sur les concepts OO avant de commencer à jouer, puis qu'ils répondent à un post-test une fois la partie terminée. De cette façon, on pourra déterminer si le jeu de rôle proposé ici a un impact sur l'enseignement de l'orienté-objet.

Il est important de préciser que ces résultats sont soumis à un biais de confirmation. En effet, dans notre expérimentation, les participants sont majoritairement des étudiants en informatique. Or, le cours d'orienté-objet donné à l'UNamur n'est pas destiné uniquement aux étudiants en informatique, on y retrouve entre autre des étudiants en mathématiques ou encore en ingénieur de gestion. Pour pallier à ce problème, il faudra réaliser en test en condition réelle avec un public plus varié.

5.5 Limites et perspectives futures

Les discussions lors de la dernière phase des expérimentations ont également soulevé quelques limites au dispositif. Cette section présentera ces différentes limites. Celles-ci seront accompagnées de différentes réflexions afin de les dépasser.

La première limite du jeu est qu'il ne faut pas demander aux étudiants d'assimiler autant de concepts orienté-objet en une fois. Une solution qui peut être mise en place facilement est de fragmenter la partie. Chaque bout de partie ne ciblera qu'un nombre réduit de concepts. L'idée est donc d'utiliser un scénario de jeu évolutif qui suivra la chronologie du cours. Ainsi, le jeu commencerait très simplement et se complexifierait et intégrerait des nouveaux concepts au fur et à mesure que le cours avance. Par exemple, lors de la première séance d'utilisation du jeu, il est demandé aux étudiants de construire leur personnage. Une fois cela fait, on leur présente les concepts de base qui sont présents dans cette phase de jeu (classe, objet,...). Puis lors des séances suivantes, le scénario se basera sur les concepts liés aux méthodes, puis aux concepts liés à l'héritage, etc. Il faut donc travailler sur un scénario qui soit divisé en plusieurs parties et où chaque partie ne se concentre que sur quelques concepts à la fois.

Une seconde limite de notre dispositif est que participer à un jeu de rôle peut refroidir les non-initiés. En effet, jouer à un jeu de rôle peut demander une grande phase d'apprentissage des règles et cette phase peut donc freiner l'envie de certains. Afin de réduire le temps consacré à l'apprentissage des règles, il faut trouver une manière simple de les présenter. Dans l'état actuel des choses, un document de quinze pages présente les règles. Cela peut être lourd pour un joueur non expérimenté. On peut imaginer réaliser une courte vidéo de cinq à dix minutes expliquant les règles pour pallier à ce problème.

Un autre problème, lié au point précédent est l'implication des joueurs. Le jeu de rôle est un exercice particulier qui demande une forte implication des joueurs afin de rentrer en immersion dans le monde et dans l'univers proposé par le MJ. Il est difficile de prédire si les étudiants seront réceptifs à ce genre de jeu.

L'inclusion des joueurs est également importante. Il ne faut pas oublier qu'au cours d'orienté-objet des étudiants de domaines différents sont présents. Par exemple, lors de l'expérimentation, on m'a reproché que le scénario était fort axé sur les combats. Étant donné la variété du public du jeu, il est difficile de déterminer les préférences de chacun. Le choix du scénario est donc

primordial pour plaire aux joueurs. Il serait donc intéressant de réfléchir à d'autres types de scénarios, comme des scénarios d'enquêtes ou d'énigmes par exemple.

La dernière limite majeure à prendre en compte est la scalabilité de notre dispositif. Comment notre dispositif d'enseignement compte gérer un groupe de vingt, trente ou encore quarante étudiants ? Une idée serait de répartir les étudiants en différents sous-groupes, de 4 à 6 joueurs afin de faire participer tous les joueurs. Mais cette répartition pose également un problème. Au delà de répartir correctement les étudiants en différents groupes, il faut trouver suffisamment de personnes capables d'endosser le rôle de MJ. Si par exemple, on a une classe de quarante élèves et qu'on les répartit en sous-groupes de cinq, il faut trouver huit encadrants de cours (professeurs ou assistants) capables de jouer le rôle de MJ. Il s'agit donc d'une réflexion à avoir avec les différents acteurs du cours d'orienté-objet afin de trouver la solution la plus adéquate pour mettre en place ce jeu en condition réelle.

6 Conclusion

Dans ce travail, nous avons présenté et créé un nouveau dispositif d'enseignement de l'orienté-objet. La conception de ce dispositif a été encadrée par une méthodologie de recherche orientée par la conception. Cette méthodologie a été décomposée en différentes phases. Chacune des phases impliquait la participation d'un ou plusieurs experts dans le domaine de l'orienté-objet. Dans un premier temps, on a mis en évidence tous les concepts orienté-objet existant. Ensuite, on a créé un listing de correspondance entre ces concepts et des éléments propres à l'univers du jeu de rôle. Ce listing a été soumis à quatre experts grâce à un questionnaire. Une fois ce listing validé par les experts, on a mis en application ces correspondances lors de la création des règles du jeu. Le document qui en ressort (Appendice E) est un mode d'emploi quant à l'utilisation du jeu lors d'un cours d'orienté-objet. Une fois ces règles créées, on les a testées lors d'une expérimentation. Les sujets de l'expérimentation étaient des assistants-chercheurs de la faculté d'informatique de l'UNamur, ainsi que des étudiants ayant déjà une expérience en orienté-objet. L'objectif de cette expérimentation était de récolter l'avis des participants, mais également de montrer les limites de ce dispositif. Il en est ressorti que le jeu était amusant et que représenter les concepts théoriques par des exemples du jeu permettait de créer une continuité et de ne pas devoir réapprendre un nouvel exemple à chaque fois qu'un nouveau concept est introduit. De plus, la liberté proposée par le jeu permettait de créer des exemples personnels et uniques pour chaque étudiant. Cependant, plusieurs limites du dispositif ont été soulignées. Ces limites permettent d'encadrer la réflexion sur l'évolution future du travail et sur la mise en pratique d'un tel dispositif. La première limite était que réunir tous les concepts dans le même jeu demandait un travail d'assimilation conséquent pour l'étudiant. Il faudra donc trouver un moyen de fractionner et de segmenter le jeu en fonction des concepts. La deuxième limite était que la présentation des règles était lourde, il faudra alléger ces règles ou trouver une autre manière de les présenter. Ensuite, on a discuté de l'implication des joueurs. Ce dispositif ne fonctionne en effet qu'avec une participation active des étudiants. Une autre limite qui a été soulignée est l'inclusion des joueurs, d'où l'importance du scénario et du rôle de MJ. Enfin, le dernier point à prendre en compte pour une évolution future est la scalabilité. Ce dispositif nécessite un test en condition réelle avec des apprenants en orienté-objet afin de confirmer ces premiers résultats. Mais ces derniers indiquent que l'on peut apporter motivation, amusement et clarté à l'enseignement de l'orienté-objet et qu'il est donc intéressant de continuer dans cette voie.

Références

[1] P. J. Burton and R. E. Bruhn, "Teaching programming in the OOP era," *ACM SIGCSE Bulletin*, vol. 35, no. 2, p. 111, 2003.

[2] R. Decker and S. Hirshfield, "The top 10 reasons why object-oriented programming can't be taught in CS 1," no. March 1994, pp. 51–55, 1994.

[3] J. Bergin, *Data Abstractions : The Object-Oriented Approach Using C++ (IBM)*. McGraw-Hill College, jan 1994.

[4] M. M. Reek, "A top-down approach to teaching programming," *ACM SIGCSE Bulletin*, vol. 27, no. 1, pp. 6–9, 1995.

[5] J. E. Sims-Knight and R. L. Upchurch, "Teaching Object-Oriented Design Without Programming : A Progress Report," *Computer Science Education*, vol. 4, no. 1, pp. 135–156, 1993.

[6] S. Hadjerrouit, "A constructivist approach to object-oriented design and programming," *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*, vol. 31, no. 3, pp. 171–174, 1999.

[7] M. Duquesnoy, G. Gilson, J. Lambert, and C. Preat, "La pédagogie du jeu," Aug 2019.

[8] M. . L. J.-M. Sanchez, Éric ; Ney, "Jeux sérieux et pédagogie universitaire : de la conception à l'évaluation des apprentissages," *Revue internationale des technologies en pédagogie universitaire / International Journal of Technologies in Higher Education*, vol. 8, no. 1-2, pp. 48–57, 2011.

[9] A. E. Rais, S. Sulaiman, and S. M. Syed-Mohamad, "Game-based approach and its feasibility to support the learning of object-oriented concepts and programming," *2011 5th Malaysian Conference in Software Engineering, MySEC 2011*, pp. 307–312, 2011.

[10] S. Abbasi, H. Kazi, and K. Khowaja, "A systematic review of learning object oriented programming through serious games and programming approaches," *4th IEEE International Conference on Engineering Technologies and Applied Sciences, ICETAS 2017*, vol. 2018-Janua, pp. 1–6, 2018.

[11] S. Cooper, W. Dann, and R. Pausch, "Alice : A 3-d tool for introductory programming concepts," *J. Comput. Sci. Coll.*, vol. 15, p. 107–116, Apr. 2000.

[12] C. Kelleher and R. Pausch, "Using storytelling to motivate programming," *Communications of the ACM*, vol. 50, no. 7, pp. 58–64, 2007.

[13] M. Kölling, "The greenfoot programming environment," *ACM Trans. Comput. Educ.*, vol. 10, Nov. 2010.

[14] L. Yan, "Teaching object-oriented programming with games," *ITNG 2009 - 6th International Conference on Information Technology : New Generations*, pp. 969–974, 2009.

[15] S. Sharma, J. Stigall, and S. Rajeev, "Game-theme based instructional module for teaching object oriented programming," *Proceedings - 2015 International Conference on Computational Science and Computational Intelligence, CSCI 2015*, pp. 252–257, 2016.

[16] Y. S. Wong, I. M. Hayati, M. Yatim, and T. W. Hoe, "A propriety game based learning mobile game to learn object-oriented programming - Odyssey of Phoenix," *Proceedings of 2017 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2017*, vol. 2018-Janua, no. December, pp. 426–431, 2017.

[17] S. Garner, P. Haden, and A. Robins, "My Program is Correct But it Doesn't Run : A Preliminary Investigation of Novice Programmers' Problems BT - Seventh Australasian Computing Education Conference (ACE2005)," vol. 42, pp. 173–180, 2005.

[18] N. Ragonis and M. Ben-Ari, "A long-term investigation of the comprehension of OOP concepts by novices," *Computer Science Education*, vol. 15, no. 3, pp. 203–221, 2005.

[19] S. Xinogalos, "Object-oriented design and programming : An investigation of novices' conceptions on objects and classes," *ACM Transactions on Computinig Education*, vol. 15, no. 3, 2015.

[20] S. Holland, R. Griffiths, and M. Woodman, "Avoiding object misconceptions," *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*, vol. 29, no. 1, pp. 131–134, 1997.

[21] G. Fitzmaurice, "Graspable user interfaces," *Citeseer*, p. 167, 1996.

[22] H. Ishii and B. Ullmer, "Tangible bits," pp. 234–241, 1997.

[23] M. S. Markova, S. Wilson, and S. Stumpf, "Tangible user interfaces for learning," *International Journal of Technology Enhanced Learning*, vol. 4, no. 3-4, pp. 139–155, 2012.

[24] O. Shaer and E. Hornecker, "Tangible User Interfaces : Past, present, and future directions," *Foundations and Trends in Human-Computer Interaction*, vol. 3, no. 1-2, pp. 1–137, 2009.

[25] S. Greenberg and C. Fitchett, "Phidgets : Easy development of physical interfaces through physical widgets," in *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, (New York, NY, USA), p. 209–218, Association for Computing Machinery, 2001.

[26] J. Underkoffler and H. Ishii, "Urp : A luminous-tangible workbench for urban planning and design," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 386–393, 1999.

[27] C. Huang, E. Y.-l. Do, and D. Gross, "MouseHaus Table : a Physical Interface for Urban Design," *16th Annual ACM Symposium on User Interface Software and Technology*, no. Figure 2, pp. 41–42, 2003.

[28] N. Couture, G. Rivière, and P. Reuter, "GeoTUI : A Tangible User Interface for Geoscience," p. 89, 2008.

[29] J. Zigelbaum, M. S. Horn, O. Shaer, and R. J. Jacob, "The tangible video editor : Collaborative video editing with active tokens," *TEI'07 : First International Conference on Tangible and Embedded Interaction*, pp. 43–46, 2007.

[30] S. Jordà, M. Kaltenbrunner, G. Geiger, and R. Bencina, "The reacTable," *International Computer Music Conference, ICMC 2005*, pp. 15–17, 2005.

[31] A. Chang, B. Resner, B. Koerner, X. Wang, and H. Ishii, "LumiTouch : An Emotional Communication Device," p. 313, 2001.

[32] C. De Raffaele, S. Smith, and O. Gemikonakli, "The Application of Tangible User Interfaces for Teaching and Learning in Higher Education," no. January, 2017.

[33] B. Schneider, P. Jermann, G. Zufferey, and P. Dillenbourg, "Benefits of a tangible interface for collaborative learning and interaction," *IEEE Transactions on Learning Technologies*, vol. 4, no. 3, pp. 222–232, 2011.

[34] T. Speelpenning, A. N. Antle, T. Doering, and E. Van Den Hoven, "Exploring how tangible tools enable collaboration in a multi-touch tabletop game," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6947 LNCS, no. PART 2, pp. 605–621, 2011.

[35] O. Zuckerman, S. Arida, and M. Resnick, "Extending tangible interfaces for education : Digital montessorl-inspired manipulatives," *CHI 2005 : Technology, Safety, Community : Conference Proceedings - Conference on Human Factors in Computing Systems*, pp. 859–868, 2005.

[36] M. S. Horn, E. T. Solovey, and R. J. Jacob, "Programming and informal science learning : Making tuis work for museums," *Proceedings of the 7th International Conference on Interaction Design and Children, IDC 2008*, pp. 194–201, 2008.

[37] F. Zamorano, C. Cortes, and M. Herrera, "A tangible user interface to facilitate learning of trigonometry," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 14, p. 152, 12 2019.

[38] D. Gajadur and G. Bekaroo, "TangiNet : A Tangible User Interface System for Teaching the Properties of Network Cables," *2nd International Conference on Next Generation Computing Applications 2019, NextComp 2019 - Proceedings*, no. September, pp. 1–6, 2019.

[39] Wikipedia contributors, "Monopoly (game) — Wikipedia, the free encyclopedia," 2020. [Online ; accessed 4-May-2020].

[40] E. van Loenen, "Entertaible : A solution for social gaming experiences," 01 2007.

[41] D. Mortel and J. Hu, "Apartgame : a multiuser tabletop game platform for intensive public use," pp. 49–52, 01 2007.

[42] A. Wilson and D. Robbins, "Playtogether : Playing games across multiple interactive tabletops," 01 2007.

[43] J. M. R. Corral], A. C. Balcells], A. M. Estévez], G. J. Moreno], and M. J. F. Ramos], "A game-based approach to the teaching of object-oriented programming languages," *Computers Education*, vol. 73, pp. 83 – 92, 2014.

[44] Sanchez and Réjane, "Recherche collaborative orientée par la conception, howpublished = `https://journals.openedition.org/educationdidactique/2288`, note = Accessed : 2020-04-23," 2017.

[45] T. Anderson and J. Shattuck, "Design-based research : A decade of progress in education research ?," *Educational Researcher*, vol. 41, no. 1, pp. 16–25, 2012.

[46] B. Liskov and J. Guttag, *Program Development in Java : Abstraction, Specification, and Object-Oriented Design*. USA : Addison-Wesley Longman Publishing Co., Inc., 1st ed., 2000.

A Listing sur les concepts Orienté-Objet

Importance	Concept	Description
1	Classe	Morceau de programme. A tout moment de l'exécution, chaque classe du programme peut avoir plusieurs objets (instance).. Deux utilités d'une classe : -Définir des collections de procédures/méthodes/opérations (abstraction procédurale) -Définir des types de données (abstraction de données)
2	Classe Concrète	Classe qui fournit une implémentation complète d'un type, peut être instancié
2	Classe Abstraite	Classe qui fournit au plus une implémentation partielle d'un type, ne peut être instancié (peut avoir des méthodes abstraites,sans implémentation)
1	Objet	Morceau de programme en cours d'exécution qui possède un état (contenant des données) et un comportement fait d'opérations (méthodes). Il s'agit d'une instance d'une classe. Les objets représente souvent des entités du monde réel. Pour créer un nouvel objet, on utilise une méthode particulière appelée constructeur + new pour référencer
3	Encapsulation	l'interface d'un objet est accessible, son implémentation est cachée. Encapsulation au niveau de classes et des packages Visibilité permettant de préciser quelles parties du codes ont accès aux variables/méthodes/classes
1	Principe de Parnas	"La définition d'une classe doit fournir à l'environnement toutes les informations nécessaires pour manipuler correctement une instance de la classe et rien d'autre."
2	Type Primitif	Type prédéfini par le langage (8 en Java)
1	Type d'objet	Un type d'objet définit un ensemble d'objet. Le programmeur peut en définir autant qu'il veut
2	Variables Primitives	Variable de type primitif contient une valeur de ce type. Toute variable peut être initialisée au moment de sa déclaration mais elle doit l'être avant son utilisation
2	Variables de Références	Une variable d'un type objet contient une référence vers un objet de ce type

1	Stack et Heap	La JVM possède 2 mémoires la stack (pile) et le heap (tas). Les variables locales sont sur le stack (peu importe si elle contient une valeur primitive ou une référence). Les objets sont dans le heap. L'instruction New permet d'allouer une partie du heap au stockage d'un objet du type indiqué, d'initialiser l'objet, d'assigner une valeur de référence et de retourner cette valeur de référence.
2	Partage de Référence	Si une variable de référence se voit donner une valeur de référence déjà assignée à une autre variable, il y a partage de référence car les deux variables référencent le même objet. Toute modification de l'objet se répercute alors chez toutes les variables.
3	Garbage Collector	Partie de la JVM chargée de libérer la mémoire (heap) occupée par les objets qui ne sont plus référencés
1	Mutabilité	Un objet est mutable si son état peut changer, sinon il est immutable. Si un objet mutable est partagé par plusieurs variables, toute modification de l'objet effectuée à partir d'une des variables est visible au travers des autres
1	Méthodes	Bout de code exécuté lorsqu'on l'invoque. On peut y passer des données sous forme de paramètres. Les méthodes doivent appartenir à une classe et définissent le comportement de l'objet.
4	Type Checking	Java est un langage typé. Le compilateur vérifie que toutes les assignations et que tous les appels sont correct du point de vue du typage. Donc, les déclarations de variables requiert une vérification du type et les méthodes doivent également indiquer les types de ses paramètres et son type de retour (et les types d'exceptions)
4	Type Safety	Plusieurs mécanismes existent pour s'assurer qu'aucune erreur de typage ne peut survenir durant l'exécution. Le type checking est un des ces mécanismes. Il y a aussi l'Automatic Storage Management et la vérification des dépassements de limites dans les tableaux
4	Automatic Storage Management	Allocation et désallocation de mémoire dans le heap. Elle s'occupe de la création d'objet via le new et de la désallocation lorsque l'objet n'est plus référencé via le garbage collector.
1	Hiérarchie	Les types non primitifs sont organisés de manière hiérarchique. Un Type donné T peut avoir plusieurs supertype. T est alors un sous type de chacuns de ses supertypes. Relation transitive : Si R est un sous type de S et S est un sous type de T alors R est un sous type de T Relation réflexive : T est sous type de lui même L'ancêtre commun de tous les objets et le super type Objects (en java)

1	Principe de Substitution (Liskov)	Principe de Substitution : Si S est un sous type de T, les objets de type S doivent pouvoir être utilisés dans tout contexte qui requiert des objets de type T.
2	Type déclaré	Type d'une variable tel qu'il est donné au moment de sa déclaration. Il est le même durant toute l'exécution
2	Type effectif	Type d'une variable à un moment donné de l'exécution. Déterminé par le type de l'objet référencé dans le Heap. Celui-ci est déterminé à la création de l'objet.
3	Casting	Permet d'indiquer le type effectif de l'objet à l'exécution. Utile pour pouvoir invoquer une méthode sur un objet dont le type déclaré l'interdit mais dont on sait que le type effectif le permet ou pour assigner à une variable une valeur dont le type déclaré l'interdit mais dont on sait que le type effectif le permet.
1	Règles des Signatures	Le sous type doit avoir toutes les méthodes du super-type et leurs signatures doivent être compatibles
1	Règles des Méthodes	Les appels aux méthodes du sous-type doivent se comporter comme des appels aux méthodes du super-type
1	Règles des Propriétés	Le sous type doit préserver toutes les propriétés qui sont vérifiées par le super-type
2	Dispatching	Permet de garantir que quand une méthode est invoqué sur un objet, il s'agit bien de la méthode du type effectif de l'objet. Chaque objet dans le heap contient une référence vers un dispatch vector qui contient une entrée pour chaque méthode d'un type d'object donné.
3	Overloading	Autorise des méthodes à avoir le même nom mais des signatures différentes. Si ambiguïté, le compilateur choisira la méthode la plus spécifique. Sinon le compilateur signale une erreur. S'applique tant aux types primitifs qu'aux types objets.
2	Type d'Objet Primitif	Chaque type primitif possède son équivalent en type Objet. Cela permet de les utiliser lorsque le contexte requiert un type Objet.
3	Autoboxing/ Unboxing	L'autoboxing est le mécanisme permettant de convertir automatiquement un type primitif dans son type objet correspondant. La conversion inverse est l'Unboxing.
1	Abstraction par spécification	La spécification est suffisante pour comprendre le comportement de la procédure, peu importe la manière dont elle est implémentée.
2	Spécification	Les abstractions sont écrites au moyen de spécifications. Elle peut être écrite dans un langage de spécification qui peut être soit formel (mathématique) soit informel (français). La spécification comprend sa signature (son nom, noms et types de paramètres, type de résultat et les exceptions renvoyés) ainsi qu'une description des effets.

1	Principe de Localité	L'implémentation d'une instruction peut être lue/écrite sans qu'il n'y ait besoin de consulter l'implémentation d'aucune autre abstraction
1	Principe de Modifiabilité	Une abstraction peut être réimplémentée sans nécessiter la réimplémentation des parties de programme qui l'utilisent
1	Abstraction Procédurale	En java, procédure = méthode statique, bout de code qui se comporte toujours de la même façon, peu importe l'objet sur laquelle elle est exécutée. Les procédures doivent être modulaire (il faut décomposer mais pas trop), simples (utilité bien définie et facilement exprimable)
2	Exceptions	Si la procédure appelée se termine normalement, elle renvoie une valeur du type de retour. Si un problème survient pendant son exécution, une exception est renvoyée. Les types d'exceptions sont organisées hiérarchiquement, soit ce sont des sous types de RuntimeException, soit ce sont des sous types d'exception
1	Abstraction de Données	S'abstraire du format et de l'interprétation des données en fournissant des opérations permettant l'accès aux donnée des objets. En java, définition d'un nouveau type au moyen d'une classe, de méthodes d'instance qui permettent l'accès aux données et de constructeurs qui permettent de créer et d'initialiser des objets. Afin d'implémenter un type, il faut choisir une représentation de données pour les instances de ce type, implémenter les constructeurs selon cette représentation, implémenter les méthodes qui modifient ou utilisent la représentation d'une instance
3	Méthodes equals, clone et toString	Le type Objet (super type de tous les objets) prédéfini les méthodes clone, equals et toString. Il se peut que l'implémentation héritée ne convient pas, il faut alors la réimplémentée. -Equals représente l'équivalence comportementale. Un objet mutable n'est égal qu'avec lui même (similaire à ==) tandis que deux objets immutables sont égaux si ils ont le même état (réimplémentation nécessaire) -Clone crée une copie d'un objet qui a le même état que celui-ci. L'implémentation par défaut crée un nouvel objet du même type et assigne à ses variables d'instances les mêmes valeurs que l'objet initial. Cependant, cette méthode peut ne pas convenir aux objets mutables car cela peut créer un partage de référence non désiré. Convient pour les types non mutables. -toString renvoie un string représentant l'état courant de l'objet ainsi qu'une indication de son type. De base, toString renvoie un hashcode, il faut donc la réimplémenter.
2	Fonction d'Abstraction	Toute implémentation d'une abstraction de donnée doit définir comment les objets de ce type sont représentés. Il s'agit d'une fonction mathématique qui permet de définir la représentation des données
2	Invariant de Représentation	Il s'agit d'une propriété que toutes les instances légitimes d'un type doit satisfaire.

1	Créateurs	Méthodes permettant de créer de nouvelles instances du type from scratch (ex : constructeur)
1	Producteurs	Méthodes permettant de prendre en entrée un objet de leur type et créent un autre objet de leur type
1	Mutateurs	Méthodes permettant de modifier les objets de leur type
1	Observateurs	Méthodes prenant en entrée des objets de leur type et retournent de objets d'autres types
3	Adéquation de Type	Un type est adéquat s'il fournit un ensemble d'opérations permettant aux utilisateurs de réaliser tout ce qu'ils désirent sur les objets de manière raisonnablement facile et efficace. -Avoir au moins 3 catégories de méthode sur les 4 (créateurs, observateurs et mutateurs si mutable et créateurs, observateurs et producteurs si immutable) -Permettre un peuplement complet du type.
4	Abstraction de l'itération	L'itérateur est une méthode ou une procédure qui fournit une manière de générer des éléments d'une collection à la demande. Les itérateurs sont une généralisation des boucles for.
4	Interface	Une classe sert à définir un type et à en donner une implémentation complète ou partielle. Tandis qu'une interface ne sert qu'à définir un type, elle ne fournit pas d'implémentation, elle ne contient que des méthodes à la fois publiques, non static et abstraites

B Listing de correspondance entre concept orienté-objet et jeu de rôle

Concept	**Description**
Classe	Les classes représentent les personnages et leur fonction. Il y a aussi des classes qui représentent les armes. Une super classe Personnage possède des sous classes Chevalier/Mage/Archer par exemple
Classe concrète	Les classes Archer, Mage et Chevalier sont des classes concrètes, elles peuvent être instanciées
Classe abstraite	La classe Personnage est une classe abstraite, il ne peut pas y avoir un joueur dont le rôle est "Personnage"
Objet	Les objets représentent les différents acteurs/personnages de la partie, mais également leurs armes. Par exemple, il y a Merlin le mage, Robin l'archer et Arthur le chevalier. Arthur est équipé d'Excalibur, son épée
Encapsulation	Le principe d'encapsulation est respecté mais implicite. Les différentes méthodes des personnages ont des visibilités
Principe de Parnas	Traduire le principe en règle du jeu : "La définition d'un rôle doit fournir toutes l'informations nécessaires pour manipuler correctement un personnage de ce rôle"
Type primitif	Les variables d'instance (nom, point de vie, point d'attaque, etc...) sont des types primitifs
Type d'objet	Les personnages et les rôles sont des types d'objets. L'arme du personnage est un type objet également
Variables primitives	Attributs du personnage (Force, dextérité, charisme, nom,...)
Variables de référence	Représentent la vie des personnages. Lorsqu'ils meurent, ils perdent leur référencement
Stack et Heap	Tableau récapitulatif de l'état des joueurs
Partage de référence	Le coffre/inventaire partagé entre les différents personnages
Garbage collector	Le garbage collector est comme la faucheuse, qui vient chercher les personnages morts (plus référencés) pour les emmener dans l'au-delà
Mutabilité	Les personnages sont mutables (leur état peuvent changer, ils peuvent perdre de la vie, gagner en force,...) tandis que les armes sont immutables
Méthodes	Les méthodes sont les actions pouvant être effectuées par les personnages
Type Checking	Présent dans les règles du jeu, un archer n'a pas le droit d'avoir une épée en tant qu'arme principale par exemple
Type Safety	Non présent
Automatic storage management	Non présent
Héritage	Personnage est le super type, les différents rôles sont les sous types. Dans les règles du jeu, le principe de substitution doit bien être respecté, ainsi que les différentes règles (signature, méthode, propriété) également
Principe de substitution (Liskov)	Traduire le principe en règle du jeu : Par exemple : Si le grand mage est un sous type de mage (un mage particulier avec plus de pouvoir) alors le grand mage doit pouvoir effectuer toutes les actions qui requiert un mage
Type déclaré et Type effectif	Ces concepts dépendent fortement du langage utilisé. Ils ne seront donc pas présents dans la première version du jeu

Casting	Des items permettent de faire du casting. Par exemple, un gant spécial permettant de tenir un bâton, un arc ou une épée et donc de donner à une épée le pouvoir d'un bâton de mage
Règle des signatures	Respectée dans les règles du jeu
Règle des méthodes	Respectée dans les règles du jeu : par exemple une méthode "marcher" de Personnage aura le même fonctionnement que la méthode "marcher" d'un chevalier
Règle des propriétés	Respectée dans les règles du jeu : par exemple une méthode "attaquer" de Personnage qui fonctionnerait que lorsqu'il y aurait un ennemi, la même méthode dans Chevalier ne fonctionnera que dans les mêmes conditions
Dispatching	Non présent
Overloading	Non présent
Type d'objet primitif	Non présent
Autoboxing/Unboxing	Non présent
Abstraction par spécification	Respectée dans les règles du jeu, l'implémentation des méthodes est cachée, le joueur doit se repérer avec les spécifications
Spécification	Décrit le comportement des actions des personnages ainsi que le contexte d'utilisation de ces méthodes, écrit en français
Principe de localité	Non présent
Principe de Modifiabilité	Non présent
Abstraction procédurale	Des méthodes seront des procédures, comme la méthodes pour avoir les règles du jeu par exemple
Exceptions	Si le joueur utilise une méthode sans y être autorisé pour une raison ou une autre, une exception est levée et annule son action/le pénalise
Abstraction de données	Un type de données FichePersonnage peut être créé assez facilement. Il représente toutes les informations qu'on a sur un personnage (comme dans un jeu de type donjons et dragons). Chaque personnage aurait un attribut qui serait de type FichePersonnage. Fiche de personnage est immutable, on le définit au début puis il reste la même jusqu'au bout.
Méthode equals, clone et toString	Non présent
Fonction d'abstraction	Non présent
Invariant de représentation	Non présent
Créateur/Constructeur	Lors de la conception du personnage, un constructeur est utilisé
Producteurs	Pour les armes (type immutable), il peut exister une méthode pour dupliquer une arme
Mutateurs	Une méthode comme "soigner" permet de modifier l'état du personnage
Observateurs	Une méthode comme getFichePersonnage() qui permet d'afficher la fiche du personnage

Adéquation de type	Présent dans les règles : l'adéquation de type nous assure qu'un personnage qui est créé est jouable (il peut attaquer, parler, interagir,...)
Abstraction de l'itération	Non présent
Interface	Non présent

C Questionnaire conception et programmation orientés objets

Questionnaire conception et programmation orientés objets

L'objectif de ce questionnaire est de tester la correspondance entre des concepts orienté-objet et des éléments d'un jeu de rôle tangible.

Le tableau de gauche présente la liste des concepts orienté-objet alors que dans celui de droite se trouvent l'ensemble des éléments d'un jeu de rôle tangible. En tant que professionnel dans le domaine de l'orienté-objet, je vous sollicite afin de vérifier si chaque concept trouve bien sa correspondance dans un élément tangible. Cette sollicitation sert la première étape de conception d'un jeu de rôle tangible pour mon mémoire de master en sciences informatiques. J'espère donc pouvoir compter sur votre participation.

L'objectif est que la correspondance entre le concept orienté-objet et le concept du jeu tangible soit la plus évidente possible. Pour ce faire, pourriez-vous écrire dans le tableau de droite, le numéro du concept orienté-objet correspondant à l'élément du jeu de rôle tangible ? N'hésitez pas à ajouter des commentaires pour chaque concept. Par exemple si vous trouvez que le concept pourrait être amélioré, modifié, si vous trouvez qu'il est source de confusion par rapport à d'autres concepts, etc. Tous les commentaires sont les bienvenus et m'aideront à améliorer mon travail.

Merci de me renvoyer ce questionnaire par mail si possible pour le vendredi 1er mai.

Je vous remercie d'avance et vous souhaite une agréable journée.

Bien à vous,

Ludovic Wasterlain

Nom :
Prénom :
Expérience en CPOO :

Concept Orienté-Objet	
Numéro	**Concept**
1	**Classe:** Morceau de programme. A tout moment de l'exécution, chaque classe du programme peut avoir plusieurs objets (instance).. Deux utilités d'une classe: -Définir des collections de procédures/méthodes/opérations (abstraction procédurale) -Définir des types de données (abstraction de données)
2	**Classe concrète:** Classe qui fournit une implémentation complète d'un type, peut être instancié

Concept de jeu de rôle	
Numéro	**Concept**
	Il existe différents rôles de héros, les chevaliers, les mages et les archers. Chaque héros possède un type d'arme propre à son rôle. **Commentaire:**
	Chaque héros peut effectuer un ensemble d'actions qu'il peut effectuer. Ils peuvent par exemple attaquer un adversaire particulier, ou encore interagir avec des villageois, … **Commentaire:**
	Contrairement aux armes qui resteront les mêmes durant toute la durée de la partie, les Héros peuvent perdre de la vie. **Commentaire:**

3	**Classe abstraite** Classe qui fournit au plus une implémentation partielle d'un type, ne peut être instancié (peut avoir des méthodes abstraites, sans implémentation)
4	**Objet:** Morceau de programme en cours d'exécution qui possède un état (contenant des données) et un comportement fait d'opérations (méthodes). Il s'agit d'une instance d'une classe. Les objets représente souvent des entités du monde réel. Pour créer un nouvel objet, on utilise une méthode particulière appelée constructeur + new pour référencer
5	**Encapsulation:** l'interface d'un objet est accessible, son implémentation est cachée. Encapsulation au niveau de classes et des packages Visibilité permettant de préciser quelles parties du codes ont accès aux variables/méthodes/classes

	Un rôle de personnage doit fournir toutes les informations nécessaires pour manipuler correctement un personnage de ce rôle **Commentaire:**
	Chaque action est accompagnée d'une description. Cette description précise ce que fait l'action en question et quand le joueur peut l'utiliser **Commentaire:**

6	**Principe de Parnas:** “La définition d’une classe doit fournir à l’environnement toutes les informations nécessaires pour manipuler correctement une instance de la classe et rien d’autre.”
7	**Type primitif:** Type prédéfini par le langage (8 en Java)
8	**Type d’objet:** Un type d’objet définit un ensemble d’objets. Le programmeur peut en définir autant qu’il veut
9	**Variable primitive:** Variable de type primitif qui contient une valeur de ce type. Toute variable peut être initialisée au moment de sa déclaration mais elle doit l’être avant son utilisation

	Représente les attributs les plus simples de chaque personnage (ses points de vie, ses points d'attaque, de défense,...) **Commentaire:**
	Les mages, archers et chevaliers doivent avoir toutes les actions communes aux héros et leur définition doivent être identique **Commentaire:**
	Si un mage, un chevalier ou un archer est un cas particulier d'un héros, alors ils doivent pouvoir effectuer toutes les actions qui requièrent un héros **Commentaire:**
	Chaque joueur incarne un héros. Soit Merlin le mage, soit Robin l’archer, soit Arthur le Chevalier. Chaque

10	**Variable de référence:** Une variable d'un type objet contient une référence vers un objet de ce type
11	**Stack et Heap** La JVM possède 2 mémoires la stack (pile) et le heap (tas). Les variables locales sont sur le stack (peu importe si elle contient une valeur primitive ou une référence). Les objets sont dans le heap. L'instruction New permet d'allouer une partie du heap au stockage d'un objet du type indiqué, d'initialiser l'objet, d'assigner une valeur de référence et de retourner cette valeur de référence.
12	**Partage de référence** SI une variable de référence se voit donner une valeur de référence déjà assignée à une autre variable, il y a partage de référence car les deux variables référencent le même objet. Toute modification de l'objet se répercute alors chez toutes les variables.

	Héros est équipé d'une arme qui lui est propre (l'épée Excalibur pour Arthur par exemple). **Commentaire:**
	Il s'agit de l'équivalent de la faucheuse, l'entité qui vient chercher les personnages morts pour les emmener dans l'au-delà **Commentaire:**
	Certaines actions ont toujours le même comportement, peu importe les circonstances **Commentaire:**
	Type d'action particulière permettant de créer un héros **Commentaire:**
	Définis l'ensemble des éléments du même genre. Par exemple, l'ensemble Arthur le chevalier, Merlin le mage et Robin le chevalier sont tous du genre Héros

13	**Garbage collector:** Partie de la JVM chargée de libérer la mémoire (heap) occupée par les objets qui ne sont plus référencés
14	**Mutabilité** Un objet est mutable si son état peut changer, sinon il est immutable. Si un objet mutable est partagé par plusieurs variables, toute modification de l'objet effectuée à partir d'une des variables est visible au travers des autres
15	**Méthodes** Bout de code exécuté lorsqu'on l'invoque. On peut y passer des données sous forme de paramètres. Les méthodes doivent appartenir à une classe et définissent le comportement de l'objet.

	Commentaire:
	Si des actions sont communes à tous les héros, alors elles auront le même fonctionnement, peu importe le rôle du héros qui l'utilise **Commentaire:**
	Au début de la partie, avant même que le joueur choisit son rôle, son personnage existe mais il fera partie de la catégorie Héros **Commentaire:**
	Une fois que le joueur a choisi son rôle, le jeu redéfinit son personnage en fonction de ce rôle **Commentaire:**
	Le fonctionnement des différentes actions des héros est caché. Le joueur n'a accès qu'à leur définition **Commentaire:**

16	**Type Checking** Java est un langage typé. Le compilateur vérifie que toutes les assignations et que tous les appels sont corrects du point de vue du typage. Donc, les déclarations de variables requièrent une vérification du type et les méthodes doivent également indiquer les types de ses paramètres et son type de retour (et les types d'exceptions)
17	**Héritage** Les types non primitifs sont organisés de manière hiérarchique. Un Type donné T peut avoir plusieurs supertype. T est alors un sous type de chacun de ses super-types. Relation transitive: Si R est un sous type de S et S est un sous type de T alors R est un sous type de T Relation réflexive: T est sous type de lui-même L'ancêtre commun de tous les objets et le super type Objects (en java)

	Ensemble d'actions ayant pour effet de modifier l'état du héros. Par exemple une méthode pour soigner le héros. **Commentaire:**
	Permets de s'assurer que tout héros créé est jouable. Il peut parler, attaquer, interagir avec son environnement,... **Commentaire:**
	Les actions des héros ont différentes visibilités qui permettent de dire aux joueurs quelles actions ils peuvent utiliser **Commentaire:**
	Les joueurs peuvent incarner un héros dont le rôle est Chevalier, Mage ou Archer car ces rôles sont complets **Commentaire:**

18	**Principe de Substitution (Liskov):** Si S est un sous type de T, les objets de type S doivent pouvoir être utilisés dans tout contexte qui requiert des objets de type T.
19	**Type déclaré:** Type d'une variable,tel qu'il est donné au moment de sa déclaration. Il est le même durant toute l'exécution
20	**Type effectif:** Type d'une variable à un moment donné de l'exécution. Déterminé par le type de l'objet référencé dans le Heap. Celui-ci est déterminé à la création de l'objet.
21	**Casting:** Permet d'indiquer le type effectif de l'objet à l'exécution. Utile pour pouvoir invoquer une méthode sur un objet dont le type déclaré l'interdit mais dont on sait que le type effectif le permet ou pour assigner à une variable une valeur dont le type déclaré l'interdit mais dont on sait que le type effectif le permet.

	Les joueurs ne peuvent pas incarner un héros sans rôle car le rôle de héros est partiel **Commentaire:**
	Une action de Héros "attaquer" qui ne fonctionnerait que lorsqu'il y a un ennemi devant fonctionnera sous les mêmes conditions si c'est un chevalier qui l'utilise **Commentaire:**
	Un item spécial permettant de changer la nature d'une arme (un arc se transforme en une épée) permettrait à un Chevalier de tenir une arme qu'il ne pourrait normalement pas **Commentaire:**
	Permets de s'assurer notamment que chaque rôle utilise bien une arme qu'il peut utiliser. Par exemple, un archer n'a pas le droit de d'avoir une épée comme arme principale

22	**Règles des signatures:** Le sous-type doit avoir toutes les méthodes du super-type et leurs signatures doivent être compatibles
23	**Règles des méthodes:** Les appels aux méthodes du sous-type doivent se comporter comme des appels aux méthodes du super-type
24	**Règles des propriétés** Le sous-type doit préserver toutes les propriétés qui sont vérifiées par le super-type

	Commentaire:
	Les héros possèdent un inventaire commun dans lequel ils pourront stocker les objets de quêtes ou des armes. **Commentaire:**
	Représente tous les attributs des héros pouvant être exprimés de manière simple, exprimé par un nombre par exemple. **Commentaire:**
	Type d'actions permettant d'avoir une information diverse sur un héros ou sur une arme. **Commentaire:**

25	**Abstraction par spécification:** La spécification est suffisante pour comprendre le comportement de la procédure, peu importe la manière dont elle est implémentée.
26	**Spécification:** Les abstractions sont écrites au moyen de spécifications. Elle peut être écrite dans un langage de spécification qui peut être soit formel (mathématique) soit informel (en français). La spécification comprend sa signature (son nom, noms et types de paramètres, type de résultat et les exceptions renvoyés) ainsi qu'une description des effets.

	Représente la vie des personnages. Tant que cet élément est toujours lié à eux ils peuvent continuer d'être utilisé. S'ils perdent la vie, ils perdent ce lien et sont donc en attente du passage de la faucheuse **Commentaire:**
	Type d'action permettant de se dupliquer. Par exemple, un ennemi capable de créer une copie identique de lui-même **Commentaire:**
	Si le joueur utilise une action sans qu'il y soit autorisé pour une raison ou une autre, celui-ci se prend une pénalité **Commentaire:**

27	**Abstraction procédurale:** En java, procédure = méthode statique, bout de code qui se comporte toujours de la même façon, peu importe l'objet sur laquelle elle est exécutée. Les procédures doivent être modulaires (il faut décomposer mais pas trop), simples (utilité bien définie et facilement exprimable)
28	**Exception:** Si la procédure appelée se termine normalement, elle renvoie une valeur du type de retour. Si un problème survient pendant son exécution, une exception est renvoyée. Les types d'exceptions sont organisés hiérarchiquement, soit ce sont des sous-types de RuntimeException, soit ce sont des sous-types d'exception

	Les héros possèdent des caractéristiques communes et des actions communes, mais ils possèdent également des caractéristiques et des actions propres à leur rôle. Les trois types de héros peuvent par exemple ramasser un objet à terre, mais seul le mage peut lancer des boules de feu. Chaque rôle (Mage, Chevalier ou Archer) est un cas particulier du rôle de Héros **Commentaire:**
	Chaque Héros possède des caractéristiques (Attaque, défense, vitesse). Ses caractéristiques sont écrites sur une fiche de personnage. Chaque héros possède une fiche de personnage unique. **Commentaire:**

29	**Abstraction de données:** S'abstraire du format et de l'interprétation des données en fournissant des opérations permettant l'accès aux données des objets. En java, définition d'un nouveau type au moyen d'une classe, de méthodes d'instance qui permettent l'accès aux données et de constructeurs qui permettent de créer et d'initialiser des objets. Afin d'implémenter un type, il faut choisir une représentation de données pour les instances de ce type, implémenter les constructeurs selon cette représentation, implémenter les méthodes qui modifient ou utilisent la représentation d'une instance.
30	**Constructeur:** Méthodes permettant de créer de nouvelles instances du type from scratch (ex: constructeur)
31	**Producteur:** Méthodes permettant de prendre en entrée un objet de leur type et créent un autre objet de leur type

	Le tableau récapitulatif de l'état de la partie. Il comporte l'état de chaque joueur, mais également des éléments qui les entoure (comme les ennemis par exemple) **Commentaire:**

32	**Mutateur:** Méthodes permettant de modifier les objets de leur type
33	**Observateur:** Méthodes prenant en entrée des objets de leur type et retournent des objets d'autres types

34	**Adéquation de type:** Un type est adéquat s'il fournit un ensemble d'opérations permettant aux utilisateurs de réaliser tout ce qu'ils désirent sur les objets de manière raisonnablement facile et efficace. Bon plan: ● Avoir au moins 3 catégories de méthode sur les 4 (créateurs, observateurs et producteurs si mutable et créateurs, observateurs et producteurs si immutable) ● Permettre un peuplement complet du type.

Commentaires généraux :

D Listing sur les concepts Orienté-Objet dans un jeu de rôle - Version corrigée par les experts

Concept	**Description**
Classe	Il existe différents types de rôle de Héros. Il y a par exemple le rôle Chevalier, Mage ou encore Archer. Différents types d'objets peuvent également être utilisés par les Héros comme par exemple différents types d'armes.
Classe concrète	Les Héros dont le rôle spécifique est Archer, Mage et Chevalier sont incarnables.
Classe abstraite	Les joueurs ne peuvent pas incarner un héros sans rôle spécifique. Le rôle de Héros n'est pas suffisant.
Objet	Les objets représente les différents acteurs/personnages de la partie, mais également leurs armes. Par exemple, il y a Merlin le mage, Robin l'archer et Arthur le chevalier. Arthur est équipé d'Excalibur, son épée.
Encapsulation	Le principe d'encapsulation est respecté mais implicite, Les différentes méthodes des personnages ont des visibilités
Principe de Parnas	Traduire le principe en règle du jeu : "La définition d'un rôle doit fournir toutes l'informations nécessaires pour manipuler correctement un personnage de ce rôle"
Type primitf	Les variables d'instance (nom, point de vie, point d'attaque, etc...) sont des types primitifs
Type d'objet	Définis les ensembles d'éléments du monde réel du même genre. L'ensemble Héros contient tout les éléments réel définit comme des Héros : Arthur le chevalier, Merlin le mage ou Robin l'archer. D'autres ensembles existent également, comme l'ensemble des items existants dans le monde. Ces ensembles peuvent contenir des sous-ensemble, comme l'ensemble des armes qui est un sous-ensemble de l'ensemble des items
Variables primitives	Attributs du personnage (Force, dextérité, charisme, nom,...)
Variable de références	Représente la vie des personnages. Lorsqu'il meurt, ils perdent leur référencement
Stack et Heap	Tableau récapitulatif de l'état des joueurs
Partage de référence	Tous les héros possèdent un inventaire en commun dans lequel ils pourront stocker les objets de quête ou des armes qu'ils ramasseront au cours de la partie. Cet inventaire est le même pour tout les héros, par conséquent, si un héros retire un objet de l'inventaire, il sera effectivement en dehors de l'inventaire pour tout les héros
Garbage collector	Le garbage collector est comme la faucheuse, qui vient chercher les personnages mort (plus référencé) pour les emmener dans l'au-delà
Mutabilité	Les personnages sont mutables (leur état peuvent changer, ils peuvent perdre de la vie, gagner en force,...) tandis que les armes sont immutables.
Méthodes	Les méthodes sont les actions pouvant être effectuée par les personnages
Type Checking	Présent dans les règles du jeu, un archer n'a pas le droit d'avoir une épée en tant qu'arme principale par exemple
Type Safety	Non présent
Automatic storage management	Non présent

Héritage	Tous les rôles spécifiques des personnages sont issus d'une racine commune, le rôle de Héros. Ils possèdent donc des caractéristiques et des actions communes à tous les Héros et également des caractéristiques et des actions plus spécifiques à leur rôle. Au sein de leur rôle, ils peuvent s'ils sont plus spécialisés dans leur discipline avoir des attributs ou actions supplémentaires. Ex : un Chevalier général, spécialisation de chevalier, possède des attributs supplémentaires (résistance à la magie,...) et possède des actions supplémentaires (attaque critique, augmenter le moral des troupes,...)
Principe de substitution (Liskov)	Traduire le principe en règle du jeu : Par exemple : Si le grand mage est un sous type de mage (un mage particulier avec plus de pouvoir) alors le grand mage doit pouvoir effectuer toutes les actions qui requiert un mage.
Type déclaré	Retrait de ce concepts du jeu car il dépend trop du langage. De plus, il est difficile de les rendre tangible.
Type effectif	Retrait de ce concepts du jeu car il dépend trop du langage. De plus, il est difficile de les rendre tangible.
Casting	Des items permettent de faire du casting. Par exemple, un gant spécial permettant de tenir un bâton, un arc ou une épée, et donc donner à une épée le pouvoir d'un bâton de mage.
Vérifications LSP	Un Chevalier général (rôle spécifique de chevalier) a bien à sa disposition toutes les capacités d'un chevalier, tout en ayant des capacités supplémentaires.
Dispatching	Non présent
Overloading	Non présent
Type d'objet primitif	Non présent
Autoboxing/Unboxing	Non présent
Abstraction par spécification	La description des différentes actions des personnages est suffisante pour comprendre leur fonctionnement et leur résultat
Spécification	Décrit le comportement des actions des personnages ainsi que le contexte d'utilisation de ces méthodes, écrit en français
Principe de localité	Non présent
Principe de Modifiabilité	Non présent
Abstraction procédurale	Des méthodes seront des procédures, comme la méthodes pour avoir les règles du jeu par exemple
Exceptions	Si le joueur utilise une méthode sans y être autorisée pour une raison ou une autre, une exception est levée et annule son action/le pénalise
Abstraction de données	Un nouveau type d'information est créé. Il s'agit de la définition d'un Héros, d'une arme,... La définition concerne les attributs et les capacités (actions disponibles) sous la forme d'une fiche de Héros, d'arme... Cette fiche contient par exemple pour un héros, les attributs : nom du personnage, ses points d'attaque, ses points de défense et ses points de vitesse. Elle contient également les actions disponibles : attaquer, défendre, courir,...
Méthode equals, clone et toString	Non présent

Fonction d'abstraction	Non présent
Invariant de représentation	Non présent
Créateur/Constructeur	Lors de la conception du personnage, un constructeur est utilisé
Producteurs	Pour les armes (type immutable), il peut exister une méthode pour dupliquer une arme
Mutateurs	Une méthode comme "soigner" permet de modifier l'état du personnage
Observateurs	Une méthode comme getFichePersonnage() qui permet d'afficher la fiche du personnage
Adéquation de type	Permets de s'assurer que tout héros créé est utilisable correctement. Un héros est utilisable s'il fournit un ensemble d'actions permettant aux joueurs de réaliser tout ce qu'il désire avec les héros de manière raisonnable et efficace
Abstraction de l'itération	Non présent
Interface	Non présent

E Règles du jeu

Un jeu de rôle comme support d'enseignement à l'orienté-objet

Description et but du "jeu"

Ce document contient le nécessaire pour utiliser un jeu de rôle comme support d'enseignement à l'orienté-objet, à savoir:

- ➢ Des règles/conseils,
- ➢ Un tableau des concepts orienté-objet et leur équivalent en élément de jeux de rôle
- ➢ Les règles d'un jeu de rôle prêt à l'emploi
- ➢ Des personnages prêts à l'emploi et un exemple de partie

Ce jeu de rôle est à intégrer dans un cours d'orienté-objet et est destiné à des apprenants en CPOO. Ce jeu servira pour illustrer les différents concepts orienté-objet lors d'un cours CPOO. Dans l'idéal, les étudiants devront d'abord utiliser le jeu, puis suivre un cours théorique présentant les concepts OO. Cependant, ce schéma peut être adapté en fonction du cours. L'enseignant est tout à fait libre d'adapter le jeu à son cours et à sa façon d'enseigner.

Ce jeu n'a pas pour but d'expliquer les différents concepts orienté-objet, mais uniquement de les illustrer. C'est pour cela qu'il doit être couplé à un cours théorique. Le cours théorique présentera les concepts de manière théorique et l'enseignant pourra ensuite illustrer ces concepts grâce à des exemples issus du jeu.

Ce jeu de rôle consiste en une version simplifiée d'un système de jeu nommé BaSIC (un système réputé pour être simple et facilement compréhensible). Grâce à ce système, même les joueurs n'ayant aucune expérience dans les jeux de rôles peuvent prendre part à une partie sans problème.

But du jeu

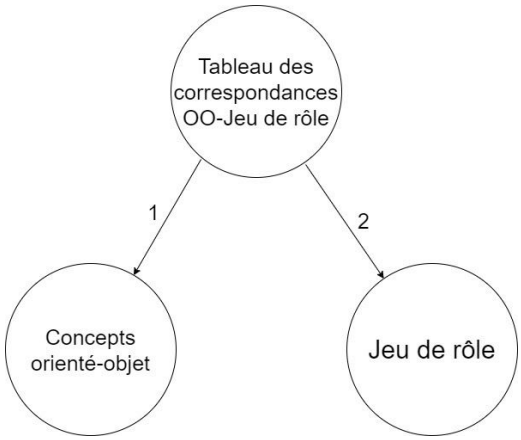
Le but est d'aider les enseignant et étudiant à mieux appréhender l'apprentissage CPOO via le jeu et une illustration de ce dernier. Les apprenant apprendront à faire le lien entre le jeu de rôle et des concepts orienté-objet. En effet, le jeu a été conçu pour être cohérent avec les différents concepts qui régissent l'orienté-objet. Utiliser des exemples issus du jeu permet aux étudiants de comprendre plus facilement ces exemples, car ils sont familiarisés avec l'univers du jeu, et permet de rendre les concepts concrets et réels. L'analogie permet de mieux comprendre et vérifier la compréhension des étudiants lors d'un cours de CPOO.

Règles générales

Avant d'utiliser le jeu comme exemple pour un cours d'orienté-objet, les étudiants doivent prendre le jeu en main. Il faut donc qu'ils se divisent en groupe de plusieurs joueurs. Un des participants doit être le meneur de jeu (de préférence, une personne avec un minimum d'expérience dans les jeux de rôles). Les autres étudiants interpréteront les différents héros. Les étudiants doivent jouer au jeu en respectant les règles du jeu de rôle (présente en annexe). Cette session de jeu peut se

Il faut également que les apprenants voient les concepts de manière théorique, que ce soit avant ou après avoir utilisé le jeu. Une fois ces concepts définis, l'enseignant doit présenter aux étudiants les liens qui existent entre le jeu de rôle qu'ils ont utilisé auparavant et les différents concepts qu'ils viennent de voir en cours. Cette séance doit avoir pour but d'illustrer les concepts qu'ils ont vu précédemment en utilisant des exemples réels issus du jeu.

Les règles décrites dans ce document permettent de comprendre comment utiliser à l'enseignant de faire les liens entre les concepts orientés-objet et les différents éléments du jeu, et explique comment les exploiter lors d'un cours.



Afin d'utiliser correctement des exemples issus du jeu, deux ressources sont disponibles en annexes de ce document: Un tableau des correspondances entre les concepts orienté-objet et leur équivalent en jeu de rôle et un livre de règles pour jouer correctement au jeu de rôle.

Le tableau des correspondances fait le lien entre les concepts et le jeu (voir schéma). Ce tableau associe un concept orienté-objet avec un élément de jeu de rôle qui le représente. Ce tableau contient les concepts principaux de l'orienté-objet(1). Il associe les concepts à un élément d'un jeu de rôle. Puis, il met en pratique les correspondances dans le jeu(2). Le document contenant les règles du jeu de rôle décrit le système de jeu de rôle utilisé (une adaptation du système de jeu BaSIC) ainsi que différentes règles à respecter afin de jouer correctement une partie.

Afin d'exploiter ces deux ressources, deux règles sont à prendre en compte:

- Tous les concepts présentés dans le tableau des correspondances sont présents dans les règles du jeu. Par conséquent, tous les concepts du tableau peuvent être illustré par des exemples du jeu;
- Le vocabulaire utilisé dans le tableau des correspondances et celui utilisé dans les règles du jeu de rôle est similaire afin de faciliter la compréhension.

Exploiter les différentes correspondances afin de s'en servir comme illustration pendant un cours peut se faire de deux manières différentes:

- Soit l'enseignant veut illustrer un concept particulier, dans ce cas il doit chercher ce concept dans le tableau des correspondances et ensuite aller chercher la mise en pratique de ce concept dans le jeu
- Soit l'enseignant prend un élément du jeu, et le déconstruit grâce au tableau des correspondances afin de montrer à ses étudiants tous les concepts représentés dans cette partie du jeu.

Exemple

Afin d'illustrer le fonctionnement du jeu dans un cours d'orienté-objet, des exemples des deux manières d'utiliser et d'exploiter le jeu sont présenté ici.

Première manière:

Disons que l'enseignant désire illustrer le concept d'héritage grâce au jeu. Dans le tableau des correspondances, le concept d'héritage est défini comme **"Tous les rôles spécifiques des personnages sont issus d'une racine commune, le rôle de Héros. Ils possèdent donc des caractéristiques, des compétences et des actions communes à tous les Héros. Mais ils possèdent en plus des compétences et des actions plus spécifiques à leur rôle. Au sein de leur rôle, ils peuvent s'ils sont plus spécialisés dans leur discipline avoir des attributs ou actions supplémentaires."**.

Cette définition comporte plusieurs éléments du jeu:

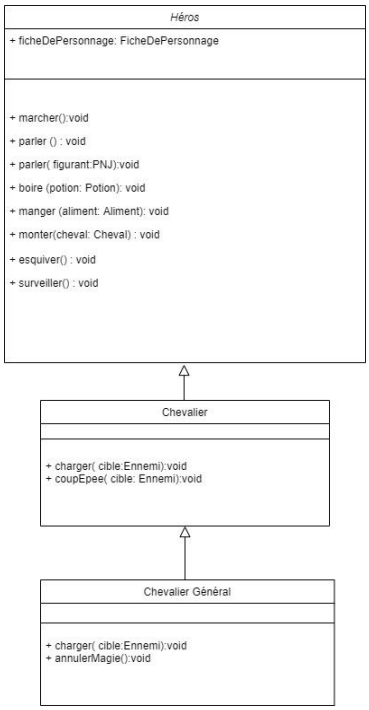
- Le rôle de Héros est la racine commune de tous les Héros;
- Caractéristiques communes, compétences communes, actions communes;
- Compétences et actions plus spécifiques en fonction du rôle (Chevalier, Archer, Mage);
- Spécialisation possible au sein du rôle (Chevalier Général, Tireur d'élite et Archimage).

Ces éléments sont bien présentés comme ceci dans les règles du jeu de rôle:

- Les caractéristiques communes sont les points de force, constitution, dextérité, charisme, intelligence et de pouvoir. Mais aussi les différentes valeurs dérivées comme les pv, les pe, le bonus de dommage et le jet de chance. Tous ces éléments sont représentés par le type FicheDePersonnage (voir abstraction de données dans le tableau de correspondances);
- Lorsqu'on regarde les compétences des héros, on remarque que trois compétences sont communes à tous les rôles:Il s'agit de la compétence esquive, vigilance et équitation;
- Les actions (méthodes) commune à tous les héros seront donc les actions basiques (marcher, parler, boire, manger,...) ainsi que les actions propres aux compétences (monter à cheval, esquiver, surveiller);
- Chaque rôle possède des compétences particulières propre à son rôle (par exemple, le chevalier possède la compétence Armes de mêlée, Armes d'Hast ou encore Charge). Par conséquent, les actions (méthodes) qu'ils possèdent en plus sont des actions propres à ces compétences (comme donner un coup d'épée ou charger);
- Une spécialisation supplémentaire existe au sein de chaque rôle. Par exemple, le Chevalier Général est une spécialisation du Chevalier. Il possède donc les mêmes compétences que le chevalier, ainsi que des compétences supplémentaire (résistance à

la magie et une version améliorée de charge). Il possède donc des actions supplémentaire (comme annuler la magie). Il possède également une version *Override* de l'action charger du Chevalier (cette version override assomme la cible).

Maintenant que tous ces éléments du jeu de rôles ont été présentés, l'enseignant peut enfin représenter le concept grâce à un diagramme de classe pour être plus visuel.



Seconde manière

L'enseignant peut très bien utiliser une partie du jeu, et la déconstruire afin de mettre en évidence les concepts qui se cachent derrière.

Disons que l'enseignant décide de prendre comme partie du jeu la création de personnage.

Dans les règles du jeu, la création du personnage se découpe en plusieurs parties, le concept associés à chaque partie peut être trouvé grâce au tableau des correspondances:

- Le joueur doit choisir un rôle, ces différents rôles sont représentés par des **classes**;
- La création d'un personnage représente l'instanciation d'une des classes de rôles, et par conséquent, la création d'un **objet.**
- Les héros possèdent différentes caractéristiques (dans le tableau des correspondances, ces caractéristiques représentent les **attributs** d'une classe. Ces attributs sont soit des **variables primitives** (comme la force, le charisme,...), soit des **variables de référence** (le nom du héros référence un objet de type String).
- Les héros sont organisés de façon hiérarchique (ils ont un chef, des caractéristiques communes, etc.). Cela représente l'**Héritage.**
- Les attributs des personnages peuvent être stockés sous la forme d'un nouveau type de données appelé FicheDePersonnage, cela représente **l'abstraction de donnée.**
- Le calcul des valeurs dérivées est toujours le même, peu importe le héros. Cela représente **l'abstraction procédurale.**

Annexes :

1. Tableau des correspondances entre jeu de rôle et orienté-objet

Concept	Description
Classe	Il existe différents types de rôle de Héros. Il y a par exemple le rôle Chevalier, Mage ou encore Archer. Différents types d'items peuvent également être utilisés par les Héros comme par exemple différents types d'armes.
Classe concrète	Les Héros dont le rôle spécifique est Archer, Mage et Chevalier peuvent être incarné par le joueur.
Classe abstraite	Les joueurs ne peuvent pas incarner un héros sans rôle spécifique. Le rôle de Héros n'est pas suffisant.
Objet	Les objets représentent les différents acteurs/personnages de la partie, mais également leurs armes. Par exemple, il y a Merlin le mage, Robin l'archer et Arthur le chevalier. Arthur est équipé d'Excalibur, son épée.
Encapsulation	Le principe d'encapsulation est respecté mais implicite. Les différentes méthodes des personnages ont des visibilités. Les méthodes publiques sont celles que le joueur peut manipuler (attaquer, parler, etc.). Les méthodes private sont celles que l'utilisateur ne peut manipuler (la méthode pour calculer les PV dans la classe FicheDePersonnage par exemple).
Principe de Parnas	Traduire en langage du jeu: "La définition d'un rôle doit fournir toutes les informations nécessaires pour manipuler correctement un personnage de ce rôle.".
Type Primitif	Les variables d'instance (pv, force, charisme, ...) sont des types primitifs.
Type d'objet	Définis les ensembles d'éléments du monde réel du même genre. L'ensemble Héros contient tous les éléments réel défini comme des Héros : Arthur le chevalier, Merlin le mage ou Robin l'archer. D'autres ensembles existent également, comme l'ensemble des items existants dans le monde. Ces ensembles peuvent contenir des sous-ensemble, comme l'ensemble des armes qui est un sous-ensemble de l'ensemble des items

Variables primitives	Attributs du personnage (force, dextérité, charisme,...)
Variables de références	Représente la vie des personnages. Lorsqu'il meurt, ils perdent leur référencement.
Stack et Heap	Tableau récapitulatif de l'état des joueurs
Partage de référence	Tous les héros possèdent un inventaire en commun dans lequel ils pourront stocker les objets de quête ou des armes qu'ils ramasseront au cours de la partie. Cet inventaire est le même pour tous les héros, par conséquent, si un héros retire un objet de l'inventaire, il sera effectivement en dehors de l'inventaire pour tous les héros
Garbage Collector	Le garbage collector est comme la faucheuse qui vient chercher les personnages morts (plus référencé) pour les emmener dans l'au-delà.
Mutabilité	Les personnages sont mutables (leur état peut changer, ils peuvent perdre de la vie, augmenter leurs compétences,...), tandis que les armes sont immutables (une fois qu'une arme est créée elle reste la même durant toute la partie)
Méthodes	Les méthodes sont les actions pouvant être effectuée par les personnages
Type Checking	Un archer n'a pas le droit d'avoir un bâton ou une épée en tant qu'arme principale. Il y a donc du type checking dans la classe archer. Le type checking se fait en fonction des compétences des Héros.
Héritage	Tous les rôles spécifiques des personnages sont issus d'une racine commune, le rôle de Héros. Ils possèdent donc des caractéristiques, des compétences et des actions communes à tous les Héros. Mais ils possèdent en plus des compétences et des actions plus spécifiques à leur rôle. Au sein de leur rôle, ils peuvent s'ils sont plus spécialisés dans leur discipline avoir des attributs ou actions supplémentaires. Ex : un Chevalier général, spécialisation de chevalier, possède des attributs supplémentaires (résistance à la

	magie,...) et possède des actions supplémentaires ou des actions de chevalier améliorée (la charge d'un chevalier général possède plus d'effet).
Principe de substitution (Liskov)	Transposée en langage du jeu. exemple: Si le grand mage est un sous-type de mage (un mage particulier avec plus de pouvoirs), alors le grand mage doit pouvoir effectuer toutes les actions qui requièrent un mage
Abstraction par spécification	La description des différentes actions des personnages est suffisante pour comprendre leur fonctionnement et leur résultat
Spécification	Le joueur décrit le comportement des actions des personnages ainsi que le contexte d'utilisation de ces actions.
Abstraction procédurale	Des méthodes du jeu sont des procédures, comme la méthode pour calculer les différentes valeurs dérivées
Exceptions	Si le joueur utilise une action sans y être autorisée pour une raison ou une autre, le MJ annule l'action et explique pourquoi au joueur.
Abstraction de données	Un nouveau type d'information est créé. Il s'agit de la définition d'un Héros, d'une arme,... La définition concerne les attributs et les capacités (actions disponibles) sous la forme d'une fiche de Héros, d'arme... Cette fiche contient par exemple pour un héros, les attributs : nom du personnage, ses différentes caractéristiques, ses points de compétences et ses valeurs dérivées.
Créateur/constructeur	Lors de la création du personnage, un constructeur est utilisé
Producteurs	Pour les armes (type immutable), il peut exister une méthode pour dupliquer une arme
Mutateurs	Une méthode comme "soigner" permet de modifier l'état du personnage

Observateurs	Une méthode comme getFichePersonnage() qui permet d'afficher la fiche du personnage
Adéquation de type	Permets de s'assurer que tout héros créé est utilisable correctement. Un héros est utilisable s'il fournit un ensemble d'actions permettant aux joueurs de réaliser tout ce qu'il désire avec les héros de manière raisonnable et efficace

2. Jeu de rôle

Description

Le jeu de rôle est une activité par laquelle une personne interprète le rôle du personnage imaginaire dans un environnement fictif. Le joueur doit interpréter le rôle de son personnage. L'interprétation du rôle implique que le joueur doit penser et agir comme son personnage. Dans ce jeu de rôle, les joueurs incarnent des Héros et ils doivent s'unir pour résoudre une quête. L'aventure est donc racontée par un maître du jeu (MJ) **et ses joueurs**. Le MJ est responsable de la fiction et supervise le jeu. Le MJ doit adapter l'aventure en fonction des choix des joueurs. Les joueurs et le MJ doivent connaître les règles du jeu de rôle avant de commencer la partie. Ces règles se trouvent en annexe.

But

Le but des joueurs d'un jeu de rôle est de vivre une aventure. racontée par le MJ **et ses joueurs.** Le joueur doit jouer un personnage, développer une personnalité, un caractère. Le point le plus important d'un jeu de rôle est la liberté des joueurs. En effet, ceux-ci sont libre de prendre les décisions qu'ils veulent, l'aventure qu'ils vivront s'adaptera en fonction de leurs choix. Le joueur doit l'incarner le temps d'une partie, penser et agir comme lui, c'est là le coeur même du jeu de rôle. Ensuite, les joueurs doivent survivre. En effet, au delà de faire vivre leur personnage, les joueurs doivent le faire survivre. Enfin, l'aventure est construite par le MJ. Le MJ est garant de la cohérence de l'histoire afin que les joueurs puissent croire en l'aventure qu'ils vivent.

Règle du Jeu de rôle(système BaSIC)

Les joueurs doivent incarner un héros, c'est la partie la plus importante d'un jeu de rôle. S'ils ne l'ont pas encore fait, ils doivent commencer par créer un héros.

Création du héros

1. Définitions

Un héros est définis par deux séries de chiffres: Ses caractéristiques et ses compétences. Les caractéristiques définissent ce qu'est le héros, tandis que les compétences définissent ce qu'il sait.

a. Les caractéristiques

Chaque héros possède 6 caractéristiques. Les valeurs de ces caractéristiques vont de 8 à 18.

- Force (FOR): Mesure de la puissance musculaire d'un personnage. Cette caractéristique donne une idée de ce qu'il peut soulever, porter, pousser ou tirer.
- Constitution (CON): Mesure la vitalité du personnage. Plus sa Constitution sera élevé, plus le héros resistera à la fatigue, aux coups, à la maladie,etc…
- Dextérité (DEX): Mesure la vivacité et la rapidité physique du héros. Cette caractéristique joue un rôle important dans les combats. Les héros possédant une dextérité élevée agissent plus souvent en premier.
- Charisme (CHA): Mesure la beauté du héros, sa capacité à s'attirer les sympathies et à charmer.
- Intelligence(INT): Mesure la capacité d'apprentissage, de mémorisation et d'analyse du personnage.
- Pouvoir(POU): Mesure la volonté du héros, sa force d'âme et sa capacité à utiliser la magie

b. Les compétences

Les compétences sont des domaines de connaissance que votre héros a eu le temps d'étudier au cours de sa vie. Les compétences se mesurent sur 100, elles seront détaillées dans le chapitre "Compétences".

2. La création en pratique

a. Etape 1: Les caractéristiques

Vous devez lancer 6 fois de suite 2d6+6 (deux dés à 6 faces +6). Puis vous répartissez les 6 résultats à votre guise parmi les 6 caractéristiques.

b. Etape 2: Les valeurs dérivées

i. **Les points de vie (PV):** mesure l'état physique du héros. Lorsqu'il est blessé, il est blessé, il perd des points de vie. S'il tombe à 0 points de vie, il est mort. Il faut donc créer un nouveau personnage. Les PV sont égaux à la CONstitution

ii. **Les points d'énergie (PE):** Les PE servent à faire de la magie. Le capital de points d'énergie est égal à votre Pouvoir (POU);

iii. **Le bonus au dommages:** Plus un personnage est grand et fort, plus il fera mal lors des combats au corps à corps. Le Bonus correspond à la FORce du personnage

Table des bonus aux dommages	
FOrce	**Bonus**
0 à 10	Aucun

11 à 14	+1d3
15 à 17	+1d6
18	+2d6

iv. **Le jet de chance:** Si les joueurs ont besoin d'un coup de pouce du destin, un jet de chance pourra être utilisé. Le jet de chance est un pourcentage, noté sur 100. Il se calcule en multipliant le POUvoir par 5.

c. Etape 3: Les compétences

Regardez la liste des rôles disponibles dans l'univers du jeu et choisissez-en une. Vous avez 300 points à répartir dans les compétences qui en découlent. Ensuite vous pourrez répartir 150 points parmi toutes les autres compétences de la liste des compétences secondaires (ou dans celles que vous avez déjà prises). Pour répartir ces points, il suffit de décider du nombre de points que vous allouez à une compétence, de l'ajouter à la base de cette dernière (la valeur entre parenthèses qui se trouvent à côté de son nom sur la liste des compétences). A sa création, un personnage ne peut pas avoir plus de 90 points dans une compétence. 1 point signifie 1% de chance de réussite dans la compétence.

Voici la liste des rôles disponibles avec leurs compétences.

- Chevalier: Vigilance, Equitation, Esquive, Athlétisme, Armes de mêlée, Armes d'Hast, Charge
- Chevalier Général (chef des chevaliers): Vigilance, Equitation, Esquive, Athlétisme, Armes de mêlée, Armes d'Hast, Charge, Résistance Magique;
- Mage: Vigilance, Equitation, Esquive, Persuasion, Sagacité, Sort de soin, Sort de feu;
- Archimage (chef des mages): Vigilance, Equitation, Esquive, Persuasion, Sagacité, Sort de soin, Sort de feu, Sort de foudre;
- Archer: Vigilance, Equitation, Esquive, Armes de tir, Armes de lancer, Cascade, Étourdissement;
- Tireur d'élite (chef des archers): Vigilance, Equitation, Esquive, Armes de tir, Armes de lancer, Cascade, Étourdissement, Tir à la tête

Pendant une aventure, il ne peut y avoir maximum qu'un seul Archimage, qu'un seul Chevalier et qu'un seul Tireur d'élite. Il n'y a pas de limites sur les autres rôles.

d. Etape 4 : Les finitions

Il faut donner vie à son personnage désormais. Première étape, lui donner un nom et un diminutif ou un surnom. Une fois nommé, il faut lui donner du relief. Les points qui suivent ne sont pas obligatoires mais permettent d'étoffer le jeu.

i. **La description:** Ajoutez des détails à votre personnage comme la couleur de vos cheveux, comment vous vous habillez, êtes vous droitier ou gaucher… Basez vous sur vos caractéristiques pour votre description. Si par exemple vous avez une TAIlle élevée et une FORce faible, c'est probablement car votre masse est due à de la graisse. Utilisez votre imagination, mais restez cohérent avec l'univers du jeu et vos caractéristiques.
ii. **Le comportement:** Trouvez un ou deux détails qui caractériseront votre personnage, comme une arachnophobie ou de l'impulsivité. Encore une fois, utilisez votre imagination, mais restez cohérent avec l'univers du jeu et vos caractéristiques.
iii. **L'histoire personnelle:** Donnez un passé à votre personnage. Où avez-vous grandi ? Avez-vous de la famille ? Vous pouvez également utiliser le passé et l'histoire du personnage pour construire son apparence. Par exemple, une cicatrice suite à un combat passé.
iv. **La motivation:** Donnez une motivation à votre personnage, une raison de se lancer dans une aventure (la gloire, la richesse,...)

Règles de base

1. Règles de bon sens

- Première règle: Une action impossible à rater réussit toujours. Par exemple, inutile de lancer les dés pour savoir si votre personnage arrive à mettre un pied devant l'autre pour marcher
- Deuxième règle: Une action impossible à réussir échoue toujours. Par exemple, inutile de lancer les dés pour savoir si le héros peut sauter d'une tour de trente mètres et rester en vie en battant des bras pour voler, c'est voué à l'échec.
- Troisième règle: Les actions non importantes ne nécessite pas un jet de dés.
- Quatrième règle: Pour tous les autres cas, faites un jet de compétence
- Cinquième règle: Lorsqu'il est impossible de faire un jet de compétence, utilisez un jet de caractéristiques.

2. Jets de compétence

Le reste des situations à résoudre se situent entre ces deux extrêmes, entre l'échec et la réussite. Pour simuler cette incertitude, on utilise un jet de dés. Les compétences sont mesurés sur 100. Un personnage avec 40 dans une compétence a en fait 40% de chance de réussir. Pour utiliser une compétence, il faut lancer 1d100 (un dé à 100 faces). Si le résultat obtenu est inférieur ou égal au score de la compétence, l'action est réussie. Sinon c'est un échec.

- **Maladresse:** Lorsque le résultat est compris entre 96 et 00, l'action est manquée, mais en plus, l'échec est grave.
- **Réussite critique:** En revanche, si le résultat est compris entre 01 et 05, l'action est réussie de manière brillante.
- **Bonus et malus:** Le meneur de jeu peut décider si une situation mérite un ajustement vers le haut ou vers le bas. Dans ces cas là, il fixe un bonus ou un malus, qui vient s'ajouter à la compétence. Voici 4 exemples de bonus et malus afin de juger si une situation nécessite un ajustement:

TABLE DE CIRCONSTANCES

Modif	Circonstance	Exemple
-20%	Très difficile	A
-10%	Difficile	B
+10%	Facile	C
+20%	Plus que facile	D

A. Tirer de nuit sur cible mobile, ou désamorcer une bombe qui va sauter d'une seconde à l'autre.
B. Tirer sur cible mobile ou de petite taille, crocheter une serrure particulièrement bien conçue.
C. Tirer sur cible immobile et de grande taille, escalader un mur avec de nombreuses prises.
D. Suivre un flâneur qui ne se méfie pas, se souvenir que la bastille a été prise un 14 juillet.

- **Expérience:** Les sept caractéristiques ne peuvent pas augmenter. En revanche, les compétences peuvent s'améliorer. Au cours de la partie, si le personnage réussit une action importante, le meneur du jeu peut l'autoriser à augmenter sa compétence de quelques points.

3. Jets de caractéristique

Certaines situations ne dépendent pas d'une compétence ou la compétence est spécifique à une autre classe. Par exemple, un personnage décide de charmer une jeune fille. Dans ce cas, aucune compétence ne s'applique. En revanche, la caractéristique CHArisme s'applique bien ici. Dans ce genre de cas, la caractéristique concernée est multiplié par un nombre entre 1 et 5. Faites un jet d'1d100 sous cette valeur, comme si c'était une compétence. Le multiplicateur dépend de la difficulté de l'action, x5 si c'est plus que facile et x1 si c'est très difficile. La plupart du temps, le multiplicateur sera un x3.

4. Blessures

Les points de vie mesurent l'état physique du personnage. Lorsqu'il est blessé, il subit des dommages. Lorsqu'il blesse un adversaire, il lui inflige des dommages. Les dommages sont donc les points de vie perdus.

- **Blessures graves:** Si un personnage perd en une seule fois la moitié du nombre actuel de ses points de vie, il est blessé grièvement. Faites un jet d'1d100 sous sa CONstitution x5 pour voir s'il supporte le choc. En cas de réussite, le personnage peut continuer à agir normalement. Si le jet est raté, le personnage perd connaissance pour un laps de temps égal à (21-CON) en minutes.
- **Mort:** Un personnage qui tombe à 0 point de vie ou moins meurt instantanément. La mort est définitive. Il faut donc recréer un nouveau personnage.

5. Cause de blessures

- **Chutes:** Si un personnage fait une chute, il faut retirer 1d6 points de vie par tranche de 3 mètres. Un jet d'athlétisme ou de cascade permet d'annuler la perte d'1d6 points. Une chute importante est presque toujours mortelle. On peut cependant utiliser un jet chance pour se raccrocher aux branches à la dernière seconde par exemple.
- **Feux:**
  - Si l'on s'en sert comme arme (boule de feu du mage ou flèche enflammée), le feu inflige 1d6 points de dommages par round. Un personnage touché doit faire un jet de chance. S'il le rate, ses vêtements prennent feu, et il subit 1d6 points de dommages par round (voir partie sur les combats), jusqu'à ce qu'il ait réussi un nouveau jet de chance, ou qu'il se soit jeté à l'eau, ou roulé par terre,etc…

- ➢ Un personnage pris dans un incendie doit réussir un jet de chance par round pour que ses vêtements ne s'enflamment pas. De plus, il risque de mourir d'asphyxie.
- ➢ Un personnage qui a perdu plus de la moitié de ses points de vie à cause du feu perd également 1d3 points de CHA.

❖ **Asphyxie/noyade:** Lorsqu'un personnage risque de s'étouffer ou de se noyer, le temps est découpé en rounds. Au premier round, le personnage doit réussir un jet de CON x10 avec 1d100. Au deuxième round, le jet est sous CON x9 et ainsi de suite jusqu'au dixième round où le jet se fait sous CON x1 (pour les rounds suivants, on reste à CON x1). Dès qu'un de ces jets est raté, le personnage subit 1d6 points de dommages. Les rounds continuent jusqu'à ce que le joueur sort de la zone dangereuse ou jusqu'à ce qu'il soit secouru.

6. Guérisons

Un personnage blessé finira par guérir avec le temps. Un personnage récupère 1d3 points de vie par semaine de jeu. S'il reste au lit, et ne se livre à aucune activité fatigante, il récupère 1d6 points de vie par semaine. Dans un hôpital, cela peut monter à 2d3 points de vie par semaine. La compétences et Potions et herbes peut également soigner. Cette compétence est expliquée dans le chapitre suivant.

Compétences

1. Compétences de rôle

Compétences commune à tous les rôles

❖ **Esquiver (25%);** L'Esquive est une compétence précieuse en combat. Lors des déclarations d'intention, un personnage qui désire esquiver une attaque l'annonce, et précise quel adversaire il souhaite esquiver. Un personnage qui esquive ne peut pas attaquer. En revanche, il peut parer l'attaque. L'Esquive peut également servir à rompre le combat: dans ce cas, le personnage fait un jet d'Esquive au début du round et, s'il le réussit, on considère qu'il s'est désengagé et n'est plus en danger (sauf, bien sûr, d'éventuels tireurs).

❖ **Vigilance (20%):** Utilisez cette compétence lorsqu'un personnage file un suspect, essaye d'écouter une conversation ou de remarquer un indice, monte la garde, est sur le point de tomber dans une embuscade... C'est une combinaison d'attention, d'ouïe aiguisée et de sixième sens.

❖ **Equitation (20%):** Cette compétence permet d'utiliser les animaux de monte, comme les chevaux. En temps normal, ce n'est pas difficile. Ne demandez un jet d'Équitation que lors des poursuites et autres activité sortant de l'ordinaire, pour vous assurer que les personnages restent en selle. L'Équitation recouvre aussi les soins routiniers à donner aux bêtes.

Compétences de Chevalier

- **Athlétisme (15%):** Cette compétence regroupe toutes les activités physiques: course, nage, saut, escalade. Ne l'utilisez que lorsque les circonstances s'y prêtent. Inutile de lancer les dés pour faire son jogging matinal. En revanche, lorsqu'un personnage est poursuivi, il peut être utile de savoir s'il court vite…
- **Armes de mêlée (25%):** Cette compétence recouvre la plupart des armes tranchantes ou contondantes qui s'utilisent à une main. Cela va du gourdin à l'épée longue en passant par la dague ou la hache.
- **Armes d'hast (20%):** Cette compétence concerne toutes les armes longues, qui s'utilisent généralement à deux mains, comme les hallebardes, les piques, les lances, etc
- **Charge (15%):** Cette compétence permet de courir vers un ennemi et de lui infliger des dégâts. Cette compétence ne fonctionne que lorsque le chevalier est à distance de la cible.

Compétences de Chevalier Général

Un chevalier général possède toutes les compétences d'un chevalier. Il possède également une compétence en plus:

- **Résistance à la magie (0%):** Cette compétence permet d'éviter un sort magique. Cela permet d'éviter, un sort direct, comme une boule de feu lancée par un mage, ou cela ne pas subir une malédiction, etc...

Le Chevalier général possède également une version améliorée d'une des compétences d'un chevalier:

- **Charge (15%):**Cette compétence permet de courir vers un ennemi et de lui infliger des dégâts. Cette compétence ne fonctionne que lorsque le chevalier est à distance de la cible. De plus, le coup infligé par le Chevalier Général assomme la cible.

Compétences de Mage

- **Persuasion (15%):** Cette compétence sert à convaincre autrui du bien-fondé de ses arguments. Elle permet de simuler des discussions qui seraient trop longues à jouer vraiment, ou peu intéressantes. C'est également elle qu'il faut utiliser si les personnages tentent de plaider leur cause devant un tribunal, de baratiner un garde pour qu'il les laisse passer, etc.
- **Sagacité (30%):** Cette compétence permet au personnage qui l'utilise d'avoir une idée de l'humeur et des motivations d'un personnage non-joueur. Le meneur de jeu n'est pas obligé d'être très précis, mais il doit donner une indication ("il a peur" suffit).
- **Sort de soin (15%):** Cette compétence permet de lancer un sort afin de soigner un personnage. Un jet réussi de Sort de soin permet de faire récupérer 1d6 points de vie (1d6+2, en cas de succès critique).S'il est raté, le soigneur peut refaire une tentative le lendemain.

- **Sort de feu (15%):** Cette compétence permet de manipuler le feu. Le joueur peut envoyer des flammes sur un ennemi. Les dommages infligés par le feu sont décrit dans la partie “Cause de blessures”

Compétence d’Archimage

Un Archimage possède toutes les compétences d’un Mage. Il possède également une compétence en plus:

- **Sort de foudre (20%):** Cette compétence permet au personnage de manipuler l’électricité. Il peut donc foudroyer des ennemis à distance. Une attaque de foudre réussi inflige 1d6+3 de dommages

L’Archimage possède également une version améliorée d’une des compétences d’un Mage:

- **Sort de soin (15%):** Cette compétence permet de lancer un sort afin de soigner un personnage. Un jet réussi de Sort de soin permet de faire récupérer 1d6 points de vie (1d6+2, en cas de succès critique).S'il est raté, le soigneur peut refaire une tentative le lendemain. De plus, cette compétence permet de réparer instantanément un problème physique grave (jambe cassée, bras arraché,...), et peut ranimer les personnages inconscients.

Compétences d’Archer

- **Armes de tir (25%):** Cette compétence permet d'utiliser arcs, arbalètes, frondes, sarbacanes... Bref, toutes les armes "balistiques".
- **Armes de lancer (20%):** Cette compétence concerne les couteaux de lancer, les javelots, mais aussi les armes improvisées, comme un bête caillou ramassé au hasard d'un chemin.
- **Étourdissement (15%):** Cette compétence permet au lanceur de tirer une flèche aveuglante, qui fait baisser la précision des ennemis.
- **Cascade (10%):** Cette compétence combine l'agilité et la souplesse du personnage. Elle recouvre tout ce qui est acrobatie, jonglage ou marcher sur un fil. C'est, par exemple, l'art d'amortir ses chutes, de se déplacer sur un toit glissant par une nuit sans lune, de sauter dans un camion en marche ou de jongler avec des torches enflammées.

Compétences de Tireur d’élite

Un Tireur d’élite possède toutes les compétences d’un Archer. Il possède également une compétence en plus:

- **Tir à la tête (25%):** Cette compétence permet au personnage de se concentrer pour infliger un coup puissant à la tête de la cible.

Le Tireur d’élite possède également une version améliorée d’une des compétences d’un Archer:

- **Étourdissement (15%):** Cette compétence permet au lanceur de tirer une flèche aveuglante, qui fait baisser la précision des ennemis. De plus, la cible prendra un round pour se remettre de cette attaque.

2. Compétences secondaire

- **Chercher (20%):** On utilise Chercher pour fouiller un endroit. Un jet réussi permet de trouver un détail, un indice... en supposant que le scénario précise qu'il y ait quelque chose à trouver, bien entendu !
- **Déguisement (10%):** Grâce à Déguisement, un personnage peut modifier son apparence. Elle a généralement deux utilités : passer inaperçu dans une foule, ou se "faire la tête"d'un individu précis. Dans le second cas, le MJ peut imposer des malus aux jets de dés, si la tentative est vraiment trop improbable (un petit japonais qui essaye de se déguiser en gros irlandais rouquin aura facilement 50 % de malus, si le MJ ne décide pas tout simplement qu'il n'a aucune chance d'y parvenir).
- **Discrétion (15%):** Demandez un jet de Discrétion lorsque les personnages ont besoin de se déplacer silencieusement (par exemple pour passer non loi d'une sentinelle). Si le jet est réussi, tout va bien. S'il est manqué, ils ont fait du bruit ou projeté une ombre voyante contre le mur...
- **Serrurerie (15%):** C'est l'art et la manière d'ouvrir une serrure sans laisser de traces. Cette compétence englobe la fabrication de clés, le perçage de coffres forts et le désamorçage de piège.
- **Dressage (15%):** Le Dressage permet de mater un animal sauvage ou de l'apprivoiser. Son utilisation la plus fréquente est de dresser des montures, mais il est tout à fait possible de l'appliquer à d'autres animaux (oiseaux, singes, chiens ...). Notez que le dressage est un processus qui prend du temps.
- **Bagarre (50%):** La Bagarre consiste à cogner à coups de pied, de tête ou de poing, de manière instinctive. N'importe qui peut le faire avec des chances de succès raisonnables, mais ce n'est pas une attaque très efficace. En effet, elle n'inflige qu'1d3 points de dommages (plus, bien sûr, le bonus aux dommages de l'attaquant). On ne peut utiliser cette compétence que pour parer une attaque de Bagarre, pour des raisons évidentes.
- **Lutte (20%):** Les personnages qui ont investi dans la compétence Lutte sont mieux entraînés que les simples bagarreurs, et savent comment être efficaces. Si, au cours d'un combat, le personnage réussit un jet de Bagarre qui est également inférieur à son score en Lutte, il inflige 2d3 points de dommages.

Combats

1. Round

Le round découpe le combat afin qu'il soit jouable. Il s'agit d'une unité de temps au cours de laquelle tous les participants peuvent agir au moins une fois. Chaque round se divise en plusieurs étapes:

- **L’initiative:** Au début de chaque round, le participant avec la plus grande valeur de sa DEXtérité commence. Puis c'est le tour du deuxième, du troisième, etc, jusqu'à celui qui a la plus mauvaise valeur. Si deux joueurs ont la même valeur, ils lancent 1d6 pour se départager. Celui qui fait le meilleur score agit avant l'autre.
- **Déclaration d’intention:** Le meneur de jeu procède à un tour de table, demandant à chaque joueur ce qu'il compte faire pendant ce round. Les actions possibles sont nombreuses : attaquer, parer, esquiver, se déplacer... Inutile de donner des réponses trop détaillées. Il suffit d'annoncer quelque chose du genre "je tape sur le gros PNJ tatoué" ou "je me cache sous la table".
- **Ordre des attaques:** À l'intérieur du round, on distingue deux passes, qui se succèdent toujours dans le même ordre. Parfois, personne ne pourra agir au cours d'une passe. Dans ce cas, ignorez-la et passez à la suivante.
    - Première passe : les personnages (PJ et PNJ) qui ont des armes à projectiles prêtes à servir agissent en premier, par ordre décroissant d'initiative.
    - Deuxième passe : une fois que ces premiers tirs ont eu lieu, on prend les personnages qui n'ont pas encore agi, toujours par ordre décroissant d'initiative. Ils peuvent attaquer, parer, se déplacer... Procédez aux jets de compétence appropriés au fur et à mesure

Lorsque tout le monde a agi, le round suivant peut commencer, et l'on revient à l'étape 1.

2. Esquiver et parer

En plus de pouvoir attaquer et se déplacer, les personnages ont deux autres possibilités: l’esquive et la parade. Si le personnage veut esquiver une attaque, il doit faire un jet de Compétence Esquiver. La parade n’est quant à elle pas une compétence. Il s’agit d’une manière d’utiliser les compétences d’attaques. Lors d'un corps à corps, un personnage qui est la cible d'une attaque peut parer avec son arme. Il fait un jet de compétence, comme pour une attaque. S'il est réussi, le coup est arrêté par l'arme, et le personnage ne subit pas de dommages. Il est possible de parer le coup d'un adversaire plus rapide que soi, puis d'attaquer le moment venu, ou le contraire. En revanche, on ne peut pas parer plusieurs fois par round.

3. Blesser l’adversaire

Il suffit de réussir un jet sous la compétence qui régit l'arme utilisée. Si l'adversaire n'esquive pas ou rate sa parade, il est blessé et perd des points de vie. Pour savoir combien, on lance un certain nombre de dés, en fonction de l'arme. Chaque arme fait un certain nombre de dés de dommages (voir la table des armes). S'il s'agit d'une arme de corps à corps, on y ajoute le bonus aux dommages de l'attaquant. Bien sûr, s'il s'agit d'une arme à distance ce bonus ne joue pas. On lance les dés pour calculer le nombre de points de dommages infligés, on y soustrait l'éventuelle protection de l'adversaire, et on retire ce qui reste de ses points de vie.

4. Réussite critique et maladresse de combat

Pour presque toutes les compétences de combat, un résultat compris entre 01 et 05 signifie que les dommages infligés par l'attaque sont plus importants que prévu. Lancez deux fois les dés de dommages de l'arme, additionnez les résultats obtenus et soustrayez le résultat aux points de vie de la cible. De plus, les attaques portées avec des armes tranchantes (épée, dague, etc.) ou des armes à feu arrivent à trouver le défaut de l'armure. La protection de cette dernière est annulée. Les conséquences d'une maladresse sont potentiellement plus variées. En voici une petite liste non limitative :

- Bagarre : la cible s'écarte à la dernière seconde. Le personnage fonce dans le mur et se blesse.
- Toutes les compétences d'armes : l'arme se brise, elle reste coincée, le personnage perd l'équilibre (et ne pourra pas attaquer au prochain round), l'attaque touche un ami au lieu de la cible...

5. Table des dommages

Armes d'hast et de mêlée		
Arme	**Compétence**	**Dommages**
Dague	Mêlée	**1d3+2**
Epée	Hast	**1d6+2**
Coup de poings	Bagarre	**1d3**

Armes de tir et de lancer		
Arme	**Compétence**	**Dommages**
Arc	Tir	**1d6+2**
Dague	Lancer	**1d3+2**

3. Personnages pré-construit

Voici quelques fiches de personnages préconstruites qui peuvent être utilisée pour jouer. Il y a une fiche par classe existante

IDENTITÉ

Nom: Uther
Classe: Chevalier
Motivation: Défendre l'honneur de sa nation

Caractéristiques et valeurs dérivées

FOR	17
CON	15
DEX	12
CHA	13
INT	12
POU	10

PV	15
PE	10
Bonus Dommage	+1d6
Chance	50

Compétences:

Vigilance	40
Equitation	45
Athlétisme	60
Arme de mêlée	40
Arme d'Hast	55
Charge	50
Lutte	50
Chercher	55
Dressage	45

IDENTITÉ

Nom: Bolvar
Classe: Chevalier Général
Motivation: Maintenir l'ordre dans le monde, quel qu'en soit le prix

Caractéristiques et valeurs dérivées

FOR	13
CON	16
DEX	13
CHA	11
INT	10
POU	9

PV	16
PE	9
Bonus Dommage	+1d3
Chance	45

Compétences:

Vigilance	30
Esquive	60
Equitation	35
Athlétisme	30
Arme de mêlée	15
Arme d'Hast	35
Résistance Magique	55
Charge	40
Bagarre	35
Chercher	55
Lutte	60

IDENTITÉ

Nom: Alanna
Classe: Mage
Motivation: Faire régner l'ordre et la justice

Caractéristiques et valeurs dérivées

FOR	11
CON	13
DEX	11
CHA	14
INT	16
POU	16

PV	13
PE	16
Bonus Dommage	+1d3
Chance	80

Compétences:

Vigilance	45
Esquive	40
Equitation	30
Persuasion	70
Sagacité	15
Sort de soin	60
Sort de feu	40
Discretion	60
Déguisement	40
Chercher	50

IDENTITÉ

Nom: Antonidas
Classe: Archimage
Motivation: Acquérir le plus de sagesse et de connaissance

Caractéristiques et valeurs dérivées

FOR	10
CON	12
DEX	13
CHA	14
INT	16
POU	14

PV	12
PE	14
Bonus Dommage	Aucun
Chance	70

Compétences:

Vigilance	35
Esquive	30
Equitation	20
Persuasion	70
Sagacité	15
Sort de soin	30
Sort de foudre	40
Sort de feu	60
Serrurerie	60
Dressage	40
Lutte	50

IDENTITÉ

Nom: Valeera
Classe: Archer
Motivation: S'enrichir et devenir célèbre

Caractéristiques et valeurs dérivées

FOR	11
CON	13
DEX	16
CHA	13
INT	15
POU	8

PV	13
PE	8
Bonus Dommage	+1d3
Chance	40

Compétences:

Vigilance	30
Esquive	55
Equitation	15
Arme de tir	60
Arme de lancer	40
Étourdissement	40
Cascade	60
Discrétion	60
Serrurerie	40
Chercher	50

IDENTITÉ

Nom: Edwin
Classe: Tireur d'élite
Motivation: Maintenir sa réputation de justicier au grand coeur

Caractéristiques et valeurs dérivées

FOR	13
CON	14
DEX	15
CHA	17
INT	11
POU	10

PV	14
PE	10
Bonus Dommage	+1d3
Chance	50

Compétences:

Vigilance	40
Esquive	30
Equitation	30
Arme de tir	60
Arme de lancer	15
Étourdissement	30
Tir à la tête	35
Cascade	60
Discretion	50
Serrurerie	65
Chercher	35

F Fiche de personnage

IDENTITÉ

Nom:
Classe:
Motivation:

Caractéristiques et valeurs dérivées

FOR	
CON	
DEX	
CHA	
INT	
POU	

PV	
PE	
Bonus Dommage	
Chance	

Compétences:

G Déroulé de l'expérimentation

Déroulé de l'expérimentation

Contexte

Au vu du contexte actuel, l'expérimentation se fera en vidéoconférence via Teams.

L'expérimentation se fera deux fois, avec trois participant à chaque fois, afin d'obtenir l'avis de 6 personnes en tout. L'expérimentation aura le même déroulé les deux fois. Ce déroulé est détaillé dans la partie scénario.

Le groupe de participants consiste en un groupe d'étudiant ayant déjà suivi le cours de CPOO à la faculté d'informatique de Namur, ainsi que des assistants en informatique également à l'université de Namur. Utiliser un groupe de participants ayant déjà des connaissances en OO permet de ne pas faire l'étape de présentation théorique des concepts OO. En effet, ce qu'il faut tester avec ce jeu, c'est l'illustration des concepts grâce au jeu. De plus, leurs retours et leurs avis s'inscrivent dans la continuité de la méthodologie utilisée.

Avant l'expérimentation, il est demandé au participant de lire le document contenant les différentes règles du jeu. Ces règles seront rappelées durant la partie si besoin.

L'expérimentation a pour but de vérifier que les concepts orientés-objet peuvent être illustrée grâce au jeu de rôle.

Scénario de l'expérimentation

1. Introduction

Le jeu que nous allons utiliser est un jeu de rôle papier. Il s'agit d'une version simplifiée d'un jeu de rôle. Ce jeu de rôle a pour but d'être utilisé comme support lors de l'enseignement de l'orienté-objet. Afin de simuler l'intégration du jeu dans un cours, cette expérimentation sera découpée en deux phases. La première consiste en une partie d'environ 1h. La seconde phase consiste en un débriefing de la partie. Le but du débriefing est de discuter des différents concepts illustrés par le jeu. Entre la première et la seconde phase, un questionnaire d'appariement sera soumis aux participants afin de tester les différentes illustrations des concepts OO sans une explication détaillée. Ce même questionnaire sera une nouvelle fois soumis aux participants après la seconde phase afin de tester l'efficacité du débriefing.

2. Partie

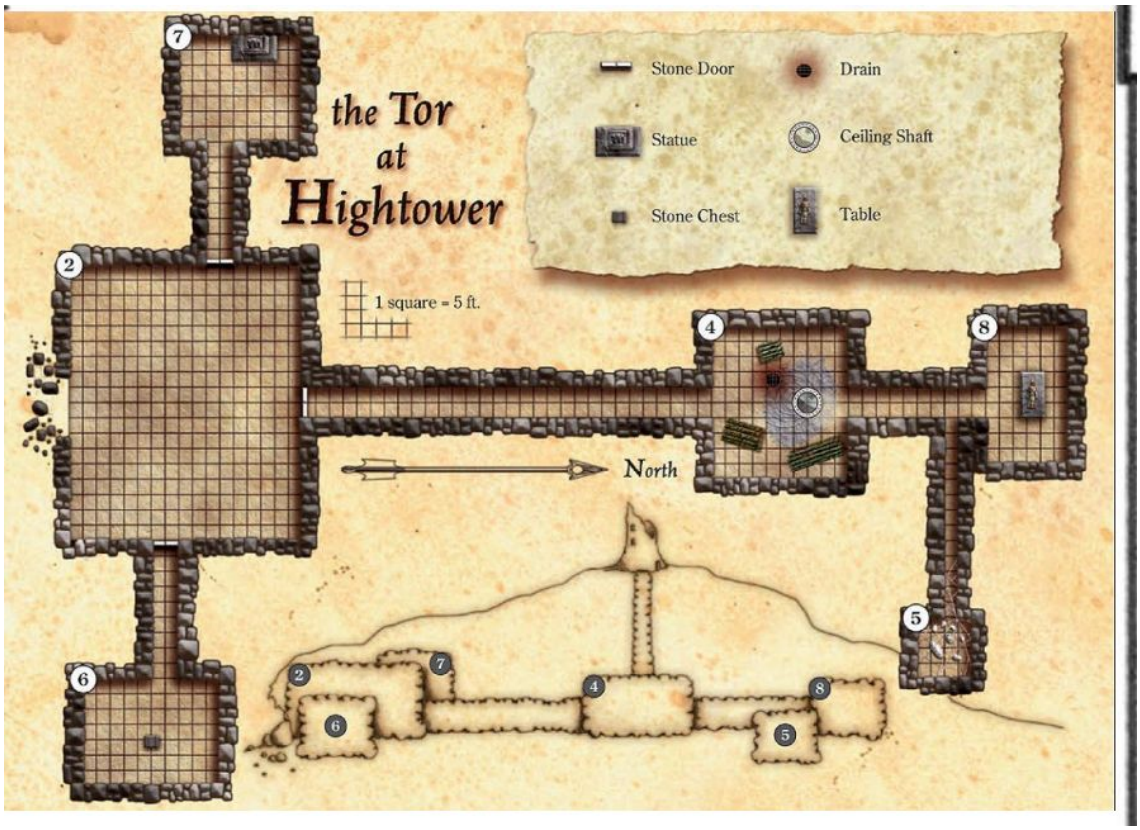
La première phase consiste en une partie de jeu de rôle. La partie durera plus ou moins 1h afin d'aborder le plus d'aspect du jeu.

Avant de commencer la partie, rappelons les règles. Les trois participants devront incarner des héros de leur choix. Ils devront donc commencer par créer un personnage, conformément aux règles du jeu. S'ils manquent d'inspiration, ils pourront utiliser un personnage conçu d'avance. Le but des joueurs est d'incarner réellement le personnage le temps d'une partie. Ils devront, le temps d'une partie penser et agir comme leur personnage.

L’histoire de la partie est prévue à l’avance et est décrite ci-dessous. Cette histoire est une adaptation de l’histoire “Un chevalier sombre et tempétueux” du jeu donjons et dragons.

➢ **LA HAUTE-TOUR**

“L’histoire commence alors qu’un groupe d’aventurier se trouve dans le désert. Autour de vous, des pics rocheux et quelques collines se dressent à l’horizon. Il pleut abondamment et vous sentez qu’un orage est proche. Vous avez été missionné par le roi en personne afin de trouver une mystérieuse tour abandonnée. La légende raconte que cette tour renferme plein de richesse. Votre royaume a besoin de cet or afin de se reconstruire après la terrible guerre qui a fait rage pendant des années. Soudain, entre deux pics, vous la voyez. Une immense tour abandonnée, le mystérieux bâtiment donne l’impression qu’il peut tomber au moindre coup de vent. Votre soif de quête et d’aventure vous guide à l’intérieur de cette étrange bâtisse. Vous remarquez un trou dans le mur, probablement causé par les ravages du temps. Vous vous y engouffrez. “



➢ **HALL PRINCIPAL DE HAUTE-TOUR**

“A l’intérieur, des peintures murales et des bas-muraux brisés suggèrent qu’elle fut autrefois décorée avec soin mais il n’en reste plus maintenant que de la poussière. Trois massives portes de pierre – une à droite, une à gauche et une droit devant, face à l’entrée – conduisent plus profondément dans le bâtiment. “

➢ **COURSE DE RATS (Aire 2)**

“Les bruits de la pluie battante, du martèlement de la grêle et du souffle du vent à l’extérieur ne sont interrompus que par les occasionnels claquements de la foudre. Quoiqu’il en soit, cette pièce semble assez solide dans l’ensemble. Même quand la foudre semble secouer la terre elle-même, rien de plus gros qu’un grain de poussière ne tombe du plafond. Les portes menant dans le pic cependant, montrent des signes de faiblesse, et tremblent à chaque coup de tonnerre. Soudain, alors que claque un coup de tonnerre particulièrement puissant, les trois portes se brisent en même temps, s’écroulant de leur chambranle sur le sol dans une belle cacophonie. Quelques secondes plus tard des sons de couinements emplissent l’air alors qu’une troupe de rats se rue depuis la porte la plus large. “

Un éclair a ouvert une crevasse dans la pièce, y entraînant l’inondation d’un nid de rats. Huit rats envahissent la chambre, à la recherche d’une voie de sortie. Rats (8) : pv 2, 2, 2, 2, 1, 1, 1, 1. Chaque rat inflige 1d6-4 de dégats et au minimum 1.

Une fois les rats vaincus, les joueurs peuvent continuer de progresser dans la tour.

➢ **CHAMBRE PRÉPARATOIRE (Aire 4)**

“Le long hall de pierre finit dans une chambre de pierre dont les seuls meubles sont des tables craquelées et travaillées sur lesquelles reposent d’anciens instruments rouillés. La pluie tombe au milieu de la salle depuis une brèche dans le plafond qui doit conduire jusqu’au sommet du pic. De cette brèche se balance une corde qui oscille. Un drain au centre du sol permet à l’eau de s’évacuer, mais de vieilles tâches suggèrent qu’il a été utilisé pour évacuer d’autres sortes de fluides il y a bien longtemps. Soudain, de l’ouverture la plus éloignée de la pièce s’avancent deux robustes humanoïdes. Chacun d’eux mesure plus de 1m80, possède des yeux sauvages et a un visage plat. Dans un grognement, ils dégainent leurs armes et chargent. “

La corde conduit effectivement au sommet du pic. Y grimper demande la réussite d’un jet de cascade. Une inspection de l’espace situé derrière l’ouverture dans le plafond, ne révèle que les restes détruits du camp hobgobelin. Cette ouverture a été creusée il y a bien longtemps par des pilleurs de tombes pour avoir accès aux richesses du pic. Hobgoblins (2): pv 10, 10 Chaque hobgoblins fait 1d6+1 de dégât.

Une fois les Hobgoblins vaincus, les héros peuvent continuer. Trésor: Des bijoux (un collier et un bracelet dans les restes du camp)

➢ **UNE TOILE BIEN ENCOMBRANTE (Aire 5)**

“Au bout de ce hall se trouve une petite chambre dont les portes de bois gisent sur le sol. L’odeur de la poussière infiltre l’air, et le son du tonnerre est ici assourdi. Des morceaux de corps sont visibles à travers le cadre de la porte, tous drapés soigneusement dans des bandes de tissus gris comme des bandelettes de momies. “

Dans le couloir, il y a une toile d’araignée, tout joueur la touchant ne peut plus bouger avant de se débarrasser de la toile.

L’araignée qui a tissé la toile est cachée dans le plafond, qui est aussi recouvert de la toile quasi-invisible. Araignée monstrueuse moyenne : pv 11 dégât: 1d6+3;

Une fois l’araignée vaincue et sa toile rendue inoffensive, les personnages peuvent inspecter la pièce. Les morceaux dans la chambre sont des corps momifiés de centaines de rats, littéralement, et de quelques humanoïdes divers. Trésor : Caché parmi les corps momifiés de la pièce se trouvent quelques armes rouillées et vêtements en lambeaux ; tous sans la moindre valeur. Sur le sol, toutefois, se trouve un saphir d’une grande valeur et une bourse contenant 75 po.

- **LE COFFRE OFFERT (Aire 6)**

“Cette salle est vide, propre de toute poussière, et seul un petit coffre de pierre est au centre sur le sol. Il semble évident que nul n’est entré dans cette chambre depuis de nombreuses années. “

Le coffre de pierre est construit dans le sol et ne peut être enlevé. Il faut donc le briser.

Piège : Le couvercle du coffre n’est pas bloqué, mais il est connecté à une trappe qui tire un barrage de dards sur quiconque se trouve dans les 1,5 m du coffre et sur chacune des cases adjacentes au coffre ou ayant un coin en commun avec la case du coffre. Un personnage doit être dans la case du coffre pour l’ouvrir, à moins qu’il ne déclare l’ouvrir avec son arme ou un bâton. Attaquer le coffre enclenche le piège de même. Le piège est des flèches qui font 1d4+1 de dégâts à la personne ouvrant le coffre. Trésor : Le coffre contient deux perles, et un petit sac contenant 100 po

- **RESTER EN TÊTE (Aire 7)**

“Le seul objet intéressant dans cette chambre est une statue à son extrémité. Elle ressemble à un sarcophage debout, sauf que la tête gravée sur le couvercle est déformée est semble avoir des racines à la place des cheveux. Une paire d’ailes de chauve-souris sort de ses côtés. Soudain, ses yeux s’ouvrent, révélant la lueur de flammes vertes, et la tête s’envole pour vous attaquer.”

Cette chambre est la tanière d’une petite vargouille qui a évidé la tête du couvercle du sarcophage afin de s’en faire un nid douillet. Petite vargouille : pv 10, dégât 1d8+1

Trésor : À l’intérieur du sarcophage maintenant sans tête, se trouve une clé qui semble ouvrir une porte dans ce bâtiment.

- **LE SOMBRE CHEVALIER (Aire 8)**

Une fois la porte ouverte grâce à la clé:

“*Une grande table de pierre domine le centre de cette chambre, et des éclats d’or brillent à travers la poussière qui recouvre le sol. Sur la table gît le corps bien préservé d’un grand humanoïde revêtu du tabard et de la ceinture d'un chevalier. Soudain le cadavre s’assoit, soulevant une vieille étoile du matin rouillée d'une main, et un javelot de l’autre. Alors qu’il se lève de la table, sa bouche s’ouvre pour laisser s’échapper un gémissement aiguë. ”*
Sombre chevalier: pv 20 arme: épée.

Trésor: Il y a un coffre à l'arrière de la salle. A l'intérieur, il y a plusieurs diamants, et un tas de pièces d'or. Cela devrait suffire pour ce donjon.

- ➢ **FIN**

L'aventure est terminée quand le Sombre Chevalier est défait et que tous les autres monstres ont étés éradiqués du massif rocheux. Une fois que ces conditions ont été réunies, l'orage à l'extérieur diminue et la tempête s'apaise. Les héros repartent vers leur royaume avec leurs trouvailles.

3. Débriefing

Cette phase consiste à discuter des différents concepts orienté-objet et également de parler des différentes illustrations fournies grâce au jeu. Lors de ce débriefing, je vais présenter deux manières d'utiliser le jeu pour illustrer. Ces deux manières seront accompagnées d'exemples de concepts. Ensuite, une discussion autour de ces manières d'utiliser le jeu commencera.

La première manière d'utiliser le jeu est de cibler un concept et de l'illustrer directement grâce à des notions du jeu. Par exemple,on peut présenter les différents concepts repris dans le tableau ci-dessous et les illustrer grâce à un exemple du jeu. Ces concepts sont différents afin de varier les exemples.

Concept	**Exemple**
Classe	Chaque rôle de héros peut être représenté par différentes classes; Il y a six classes de héros jouable: Archer, Tireur d'élite, Mage, Archimage, Chevalier et Chevalier Général. Chacune de ces classes contient l'ensemble des attributs propres à un héros (Charisme, Force, pv,...) et également une collection de méthode/actions pouvant être effectué en fonction de ces compétences. Les armes peuvent également être représentées en différentes classes: Dague, épée et arc. Chacune de ces armes contient un attribut représentant les dommages de l'arme.
Mutabilité	Les classes représentant les rôles de héros sont mutables. En effet, son état peut changer au cours d'une partie: Il peut perdre de la vie ou gagner des points de compétences par exemple. Mais malgré ces changements le héros restera le même.

	Tandis que les classes représentant les types d'armes sont immutables. En effet, les armes ne peuvent pas changer d'état au cours de la partie. Une fois définie, une arme aura toujours le même nom et infligera toujours le même nombre de dommages.
Méthodes	Toutes les actions pouvant être effectuée par les héros sont les méthodes appartenant à la classe du héros en question. Les actions sont définies par les compétences du rôle du héros. Certaines actions sont disponibles pour tous les héros (Parler, marcher, monter à cheval, esquiver,...), tandis que d'autres sont propres au rôle du joueur (Sort de soin pour le mage, charger pour le chevalier,...)
Abstraction par spécification	Lorsque le joueur décide de faire une action, celui-ci la décrit un minimum au MJ afin que tous les deux soit d'accord sur le comportement et le but de l'action. Cette description donné par le joueur représente la spécification de l'action. Cette description est nécessaire pour comprendre le comportement de l'action, peut importe l'action est exécutée. Par exemple, si un joueur décide de marcher en direction d'une auberge, le joueur ne va pas commencer à décrire toutes l'exécution de l'action (il ne vas pas dire je mets le pied droit en avant, puis le gauche, etc jusqu'à l'auberge), il va juste dire je décide d'aller jusqu'à l'auberge afin de prendre un repas par exemple.
Héritage	Les différentes classe de rôles de héros peuvent se représenter hiérachiquement. Les héros, peu importe leur rôle, possèdent des éléments en communs (actions, attributs). Ils ont donc une racine commune, qui est représenté par une classe abstraite Héros. Cette classe est le super-type de tous les autres rôles. Les caractéristiques communes sont les points de force, constitution, dextérité, charisme, intelligence et de pouvoir. Mais aussi les différentes valeurs dérivées comme les pv, les pe, le bonus de

	dommage et le jet de chance. Tous ces éléments sont représentés par le type FicheDePersonnage (voir abstraction de données dans le tableau de correspondances); Lorsqu'on regarde les compétences des héros, on remarque que trois compétences sont communes à tous les rôles:Il s'agit de la compétence esquive, vigilance et équitation; Les actions (méthodes) commune à tous les héros seront donc les actions basiques (marcher, parler, boire, manger,...) ainsi que les actions propres aux compétences (monter à cheval, esquiver, surveiller); Chaque rôle possède des compétences particulières propre à son rôle (par exemple, le chevalier possède la compétence Armes de mêlée, Armes d'Hast ou encore Charge). Par conséquent, les actions (méthodes) qu'ils possèdent en plus sont des actions propres à ces compétences (comme donner un coup d'épée ou charger); Une spécialisation supplémentaire existe au sein de chaque rôle. Par exemple, le Chevalier Général est une spécialisation du Chevalier. Il possède donc les mêmes compétences que le chevalier, ainsi que des compétences supplémentaire (résistance à la magie et une version améliorée de charge). Il possède donc des actions supplémentaire (comme annuler la magie). Il possède également une version *Override* de l'action charger du Chevalier (cette version override assomme la cible).

Lors de la présentation, une définition théorique du concept sera rappelé aux participants.

La seconde manière d'utiliser le jeu est de sélectionner une phase de jeu et de la décortiquée afin de souligner les différents concepts orienté-objet présent. Par exemple, la

phase de jeu correspondante à la création de personnage peut être décrite grâce aux concept ci-dessous:

Dans les règles du jeu, la création du personnage se découpe en plusieurs parties, le concept associés à chaque partie peut être trouvé grâce au tableau des correspondances:

- ➢ Le joueur doit choisir un rôle, ces différents rôles sont représentés par des **classes**;
- ➢ La création d'un personnage représente l'instanciation d'une des classes de rôles, et par conséquent, la création d'un **objet.**
- ➢ Les héros possèdent différentes caractéristiques (dans le tableau des correspondances, ces caractéristiques représentent les **attributs** d'une classe. Ces attributs sont soit des **variables primitives** (comme la force, le charisme,...), soit des **variables de référence** (le nom du héros référence un objet de type String).
- ➢ Les héros sont organisés de façon hiérarchique (ils ont un chef, des caractéristiques communes, etc.). Cela représente l'**Héritage.**
- ➢ Les attributs des personnages peuvent être stockés sous la forme d'un nouveau type de données appelé FicheDePersonnage, cela représente **l'abstraction de donnée.**
- ➢ Le calcul des valeurs dérivées est toujours le même, peu importe le héros. Cela représente **l'abstraction procédurale.**

Ensuite, une fois ces deux manières présentées, une discussion commencera. Cette discussion a plusieurs buts: Récolter l'avis des participants et dégager les tendances vis à vis du jeu. Cette discussion étant assez libre, elle ne sera pas décrite dans ce document.

Afin de récolter les retours des participants, ils devront répondre à un questionnaire après la deuxième phase. Le questionnaire a pour but de vérifier que les différents concepts OO sont correctement illustrés et utilisés dans le jeu.

H Questionnaire d'appariement (expérimentation)

Concept Orienté-Objet	
Numéro	**Concept**
1	**Classe:** Morceau de programme. A tout moment de l'exécution, chaque classe du programme peut avoir plusieurs objets (instance).. Deux utilités d'une classe: -Définir des collections de procédures/méthodes/opérations (abstraction procédurale) -Définir des types de données (abstraction de données)
2	**Classe concrète:** Classe qui fournit une implémentation complète d'un type, peut être instancié
3	**Classe abstraite** Classe qui fournit au plus une implémentation partielle d'un type, ne peut être instancié (peut avoir des méthodes abstraites,sans implémentation)

Concept de jeu de rôle	
Numéro	**Concept**
	Tous les rôles spécifiques des personnages sont issus d'une racine commune, le rôle de Héros. Ils possèdent donc des caractéristiques, des compétences et des actions communes à tous les Héros. Mais ils possèdent en plus des compétences et des actions plus spécifiques à leur rôle. Au sein de leur rôle, ils peuvent s'ils sont plus spécialisés dans leur discipline avoir des attributs ou actions supplémentaires. Ex : un Chevalier général, spécialisation de chevalier, possède des attributs supplémentaires (résistance à la magie,...) et possède des actions supplémentaires ou des actions de chevalier améliorée (la charge d'un chevalier général possède plus d'effet).
	Définis les ensembles d'éléments du monde réel du même genre. L'ensemble Héros contient tous les éléments réel défini comme des Héros : Arthur le chevalier, Merlin le mage ou Robin l'archer. D'autres ensembles existent également, comme l'ensemble des items existants dans le monde. Ces ensembles peuvent contenir des sous-ensemble, comme l'ensemble des armes qui est un sous-ensemble de l'ensemble des items

4	**Objet:** Morceau de programme en cours d'exécution qui possède un état (contenant des données) et un comportement fait d'opérations (méthodes). Il s'agit d'une instance d'une classe. Les objets représente souvent des entités du monde réel. Pour créer un nouvel objet, on utilise une méthode particulière appelée constructeur + new pour référencer
5	**Encapsulation:** l'interface d'un objet est accessible, son implémentation est cachée. Encapsulation au niveau de classes et des packages Visibilité permettant de préciser quelles parties du codes ont accès aux variables/méthodes/classes
6	**Principe de Parnas:** "La définition d'une classe doit fournir à l'environnement toutes les informations nécessaires pour manipuler correctement une instance de la classe et rien d'autre."
7	**Type primitif:** Type prédéfini par le langage (8 en Java)

	Représente la vie des personnages. Lorsqu'il meurt, ils perdent leur référencement.
	L'état des personnages peut changer, ils peuvent perdre de la vie, augmenter leurs compétences,...), tandis que l'état des armes ne peut changer(une fois qu'une arme est créée elle reste la même durant toute la partie)
	Les joueurs ne peuvent pas incarner un héros sans rôle spécifique. Le rôle de Héros n'est pas suffisant.
	Le joueur décrit le comportement des actions des personnages ainsi que le contexte d'utilisation de ces actions.
	La faucheuse qui vient chercher les personnages morts (plus référencé) pour les emmener dans l'au-delà.
	Les différentes méthodes des personnages ont des visibilités. Les méthodes publiques sont celles que le joueur peut manipuler, les actions qu'il peut effectuer (attaquer, parler, etc.). Les méthodes private sont celles que l'utilisateur ne peut manipuler (la méthode pour calculer les PV dans la classe)
	Tous les héros possèdent un inventaire en commun dans lequel ils pourront stocker les objets de quête ou des armes qu'ils ramasseront au cours de la partie. Cet inventaire est le même pour tous les héros, par conséquent,

8	**Type d'objet:** Un type d'objet définit un ensemble d'objets. Le programmeur peut en définir autant qu'il veut
9	**Variable primitive:** Variable de type primitif qui contient une valeur de ce type. Toute variable peut être initialisée au moment de sa déclaration mais elle doit l'être avant son utilisation
10	**Variable de référence:** Une variable d'un type objet contient une référence vers un objet de ce type
11	**Stack et Heap** La JVM possède 2 mémoires la stack (pile) et le heap (tas). Les variables locales sont sur le stack (peu importe si elle contient une valeur primitive ou une référence). Les objets sont dans le heap. L'instruction New permet d'allouer une partie du heap au stockage d'un objet du type indiqué, d'initialiser l'objet, d'assigner une valeur de référence et de retourner cette valeur de référence.

	si un héros retire un objet de l'inventaire, il sera effectivement en dehors de l'inventaire pour tous les héros
	Il s'agit des actions pouvant être effectuée par les personnages (parler, attaquer,...)
	La description des différentes actions des personnages est suffisante pour comprendre leur fonctionnement et leur résultat
	Attributs du personnage (force, dextérité, charisme,...)
	Tableau récapitulatif de l'état des joueurs
	les valeurs des variables d'instance (pv, force, charisme, ...) sont de ce type
	Des méthodes du jeu sont des procédures, comme la méthode pour calculer les différentes valeurs dérivées
	Une méthode comme "soigner" permet de modifier l'état du personnage
	Les Héros dont le rôle spécifique est Archer, Mage et Chevalier peuvent être incarné par le joueur.
	Si le joueur utilise une action sans y être autorisée pour une raison ou une autre, le MJ annule l'action et explique pourquoi au joueur.

12	**Partage de référence** SI une variable de référence se voit donner une valeur de référence déjà assignée à une autre variable, il y a partage de référence car les deux variables référencent le même objet. Toute modification de l'objet se répercute alors chez toutes les variables.
13	**Garbage collector:** Partie de la JVM chargée de libérer la mémoire (heap) occupée par les objets qui ne sont plus référencés
14	**Mutabilité** Un objet est mutable si son état peut changer, sinon il est immutable. Si un objet mutable est partagé par plusieurs variables, toute modification de l'objet effectuée à partir d'une des variables est visible au travers des autres
15	**Méthodes** Bout de code exécuté lorsqu'on l'invoque. On peut y passer des données sous forme de paramètres. Les méthodes doivent appartenir à une classe et définissent le comportement de l'objet.

	"La définition d'un rôle doit fournir toutes les informations nécessaires pour manipuler correctement un personnage de ce rôle.".
	Si l'archimage est un sous-type de mage (un mage particulier avec plus de pouvoirs), alors le grand mage doit pouvoir effectuer toutes les actions qui requièrent un mage
	Une méthode comme getFichePersonnage() qui permet d'afficher la fiche du personnage
	Un archer n'a pas le droit d'avoir un bâton ou une épée en tant qu'arme principale. Cette vérification est faite grâce aux compétences
	Il existe différents types de rôle de Héros. Il y a par exemple le rôle Chevalier, Mage ou encore Archer. Différents types d'items peuvent également être utilisés par les Héros comme par exemple différents types d'armes.
	Type d'action particulière permettant de créer un héros (au début de la partie)
	Permets de s'assurer que tout héros créé est utilisable correctement. Un héros est utilisable s'il fournit un ensemble d'actions permettant aux joueurs de réaliser tout ce qu'il désire avec les héros de manière raisonnable et efficace

16	**Type Checking** Java est un langage typé. Le compilateur vérifie que toutes les assignations et que tous les appels sont corrects du point de vue du typage. Donc, les déclarations de variables requièrent une vérification du type et les méthodes doivent également indiquer les types de ses paramètres et son type de retour (et les types d'exceptions)
17	**Héritage** Les types non primitifs sont organisés de manière hiérarchique. Un Type donné T peut avoir plusieurs supertype. T est alors un sous type de chacun de ses super-types. Relation transitive: Si R est un sous type de S et S est un sous type de T alors R est un sous type de T Relation réflexive: T est sous type de lui-même L'ancêtre commun de tous les objets et le super type Objects (en java)
18	**Principe de Substitution (Liskov):** Si S est un sous type de T, les objets de type S doivent pouvoir être utilisés dans tout contexte qui requiert des objets de type T.

	Pour les armes (type immutable), il peut exister une méthode pour dupliquer une arme
	Un nouveau type d'information est créé. Il s'agit de la définition d'un Héros, d'une arme,... La définition concerne les attributs et les capacités (actions disponibles) sous la forme d'une fiche de Héros, d'arme... Cette fiche contient par exemple pour un héros, les attributs : nom du personnage, ses différentes caractéristiques, ses points de compétences et ses valeurs dérivées.
	Représente les différents acteurs/personnages de la partie, mais également leurs armes. Par exemple, il y a Merlin le mage, Robin l'archer et Arthur le chevalier. Arthur est équipé d'Excalibur, son épée.

19	**Abstraction par spécification:** La spécification est suffisante pour comprendre le comportement de la procédure, peu importe la manière dont elle est implémentée.
20	**Spécification:** Les abstractions sont écrites au moyen de spécifications. Elle peut être écrite dans un langage de spécification qui peut être soit formel (mathématique) soit informel (en français). La spécification comprend sa signature (son nom, noms et types de paramètres, type de résultat et les exceptions renvoyés) ainsi qu'une description des effets.
21	**Abstraction procédurale:** En java, procédure = méthode statique, bout de code qui se comporte toujours de la même façon, peu importe l'objet sur laquelle elle est exécutée. Les procédures doivent être modulaires (il faut décomposer mais pas trop), simples (utilité bien définie et facilement exprimable)

22	**Exception:** Si la procédure appelée se termine normalement, elle renvoie une valeur du type de retour. Si un problème survient pendant son exécution, une exception est renvoyée. Les types d'exceptions sont organisés hiérarchiquement, soit ce sont des sous-types de RuntimeException, soit ce sont des sous-types d'exception
23	**Abstraction de données:** S'abstraire du format et de l'interprétation des données en fournissant des opérations permettant l'accès aux données des objets. En java, définition d'un nouveau type au moyen d'une classe, de méthodes d'instance qui permettent l'accès aux données et de constructeurs qui permettent de créer et d'initialiser des objets. Afin d'implémenter un type, il faut choisir une représentation de données pour les instances de ce type, implémenter les constructeurs selon cette représentation, implémenter les méthodes qui modifient ou utilisent la représentation d'une

	instance.
24	**Constructeur:** Méthodes permettant de créer de nouvelles instances du type from scratch (ex: constructeur)
25	**Producteur:** Méthodes permettant de prendre en entrée un objet de leur type et créent un autre objet de leur type
26	**Mutateur:** Méthodes permettant de modifier les objets de leur type
27	**Observateur:** Méthodes prenant en entrée des objets de leur type et retournent des objets d'autres types

28/	**Adéquation de type:** Un type est adéquat s'il fournit un ensemble d'opérations permettant aux utilisateurs de réaliser tout ce qu'ils désirent sur les objets de manière raisonnablement facile et efficace. Bon plan: ● Avoir au moins 3 catégories de méthode sur les 4 (créateurs, observateurs et producteurs si mutable et créateurs, observateurs et producteurs si immutable) ● Permettre un peuplement complet du type.

Commentaires généraux :