UNIVERSITÉ DU QUÉBEC À MONTRÉAL

GAME GENESIS

UN PROFIL UML POUR LA CONCEPTION DE JEUX VIDÉOS

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

ELLEN HAAS

AOÛT 2019

TABLE DES MATIÈRES

LISTE DES FIGURES

Figure Page

LISTE DES TABLEAUX

Tableau Page

RÉSUMÉ

INTRODUCTION

Le développement de jeux vidéos

Les métiers impactés

Les outils de travail

Etapes de développement d'un jeu vidéo

Focus sur Breakthrough et Conception

Les livrables du développement

CHAPITRE I

LE DÉVELOPPEMENT DE JEUX VIDÉOS

Trouver une manière d'introduire le chapitre

1.1 Les étapes de création d'un jeu vidéo

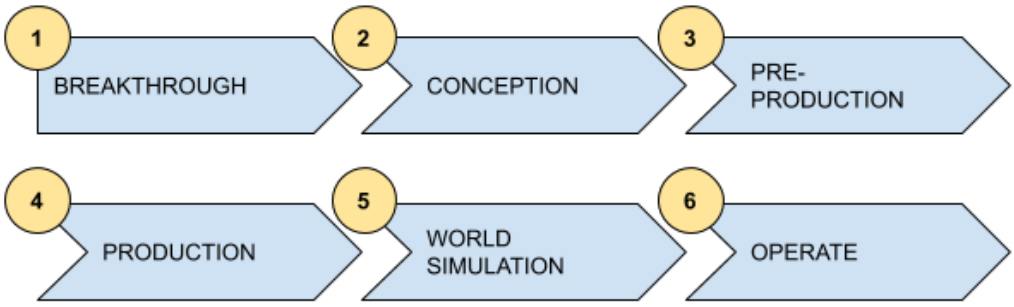

Figure 1.1: Étapes de création d'un jeu vidéo.

Les étapes de développement d'un jeu vidéo ne sont pas immuables et dépendent de l'entreprise, du domaine d'activité ou des collaborateurs impliqués dans le projet. La figure **??** présente une liste non exhaustive de ces étapes, telle que présentée par Mathieu Nayrolles lors d'un séminaire au LATECE (Laboratoire de recherche sur les Technologies du Commerce Électronique) de l'UQAM, le 10 avril 2019.

1.1.1 *Breakthrough*

Une étape de *Breakthrough* permet de réunir une équipe afin d'effectuer des recherches et des explorations sur des nouvelles mécaniques ou des nouvelles technologies afin de créer du nouveau contenu. Cette étape est optionnelle dans un projet. Deux exemples :

— Une percée technologique, par ex., Google lance Stavia, une plateforme en ligne de jeux vidéos sous forme de catalogue et jouable à 100% en ligne sans installation en local ;

— Une percée de Gameplay, par ex., naissance du mode Battle Royale.

1.1.2 Conception

Un document de conception est défini afin d'identifier l'environnement, la faisabilité et l'intérêt commercial du jeu. C'est durant cette étape que les *Game designers* définissent et précisent l'univers, les mécaniques et le déroulement du jeu vidéo en question. Cette étape est majoritairement gérée par les *Game designers* appuyés par les équipes des autres corps de métier. Dans des projets à financements externes, cette étape est cruciale : elle permettra de présenter le projet à un studio avec une première maquette présentant les fondements du jeu.

1.1.3 Pré-production

Durant cette étape, des prototypes sont développés afin de créer une version minimale du jeu. Ces prototypes permettent d'avoir un aperçu jouable des concepts définis durant la phase précédente.

Un prototype est une coupe verticale de tout le système qui permet de valider ou redéfinir les concepts. Une fois le design bien défini, le prototype plus proche de ce

que donnerait le jeu final et la faisabilité du projet confirmée, il est alors possible de rechercher les financements et les ressources nécessaires à la production. Cette étape est gérée par tous les corps de métier dans un studio de développement, tous les aspects du jeu devant être représentés afin de montrer tout le potentiel du prototype.

1.1.4 Production

Une fois les fonds levés et les ressources humaines attitrées au projet, il est possible de procéder à la production du jeu vidéo. Tous les corps de métier sont représentés et le jeu est développé sous tous ses aspects et dans son intégralité. La plupart du temps le développement est découpé en plusieurs itérations. Chacune d'entre elles permet de vérifier que le jeu respecte bien les concepts définis plus tôt. C'est également à ce moment là que les dernières modifications sont apportées aux concepts afin qu'ils respectent la vision du *Game designer* et génèrent la bonne expérience. Dans le cas où le projet rencontre des contraintes supplémentaires (temps, argent, plateforme, etc.) les concepts peuvent également être revus. Généralement, durant cette étape, le marketing et la publicité autour du jeu commencent à prendre place afin d'informer le public et d'estimer l'impact que peut avoir le jeu.

1.1.5 *World Simulation*

Lors de la *World simulation*, tous les éléments du jeu sont testés et passés au crible. On vérifie que les éléments de jeu sont correctement modélisés, que les sons correspondent bien aux éléments, que les personnages correspondent à ceux décrits dans les documents de conception, etc. Plusieurs questions se posent à ce moment là.

— Est-ce que les éléments interagissent bien entre eux ?

— Est-ce que l'univers de jeu est cohérent de bout en bout ?

— Est-ce que le *gameplay* est fluide et intuitif ?

— Est-ce que l'environnement de jeu est réellement tel que décrit par le *Game designer* ?

— Est-ce que la bande sonore ou la modélisation graphique génèrent bien les émotions attendues chez le joueur ?

1.1.6 *Operate*

Le jeu est maintenant produit et commercialisé. Une quantité importante de données est ainsi générée. Des bogues peuvent être remontés et corrigés dans des cas de figure particuliers ou inédits non couverts par l'étape de *World simulation*. Des ajustements mineurs peuvent être faits en fonction des besoins ou des demandes des joueurs. Le jeu prend alors toute sa dimension et toute sa vie à travers les joueurs.

Une fois le jeu bien en place et les étapes de corrections et ajustements passées, il est possible d'intégrer du nouveau contenu au jeu en repassant par les étapes précédentes. Ce nouveau contenu est généralement intégré au jeu sous forme de mises à jours (gratuites) ou de *Downloadable Content* (DLC) (payantes).

1.2 Exploitation et exploration

1.2.1 Exploitation

Le travail d'exploitation consiste à produire une suite ou un nouveau jeu en utilisant des technologies (moteur, plateformes, etc) ou un *gameplay* déjà existant afin de recréer un jeu ou produire du contenu additionnel. Ceci peut être fait dans un but de fidéliser une clientèle déjà existante en ajoutant du contenu additionnel

à un jeu, à offrir une expérience similaire avec des technologie plus récentes (ex : FIFA) ou à offrir une suite à un jeu ayant déjà connu du succès (ex : Dark Souls).

L'exploitation est une part importante du travail d'un studio de développement. De nombreux jeux vidéos récents sont fondés sur de l'exploitation de jeux précédents, autant au niveau du *gameplay* qu'au niveau des concepts fondamentaux de jeux précédents. C'est le cas de grosses productions de franchises comme les jeux de EA sports (FIFA, NHL, NBA Live, Madden), les jeux d'action *role-play* de FromSoftware (série des Dark Souls), les jeux d'action aventure de Rockstar (série des GTA) ou les jeux de simulation de Maxi/EA Games (série les Sims).

1.2.2 Exploration

L'exploration, ou l'innovation, dans le monde du jeu vidéo est essentielle au développement de nouveaux concepts de *gameplay* mais également au développement de nouvelles technologies. La nouveauté est un enjeu essentiel afin d'attirer toujours plus de joueurs. L'investissement dans l'exploration est donc très important pour un studio de développement. De la recherche de nouveaux concepts de jeux, de nouveaux types de *gameplay*, de nouvelles technologies à intégrer ou de la création de nouveaux moteurs de jeux, l'exploration est devenue un facteur essentiel au domaine du jeu vidéo et à son expansion.

1.2.3 L'équilibre entre exploitation et exploration

Dans leur article, Parmentier *et al.* [**?**] explorent la capacité des studios à concilier ces deux activités. Ils présentent les enjeux de chacune d'entre elles et leur importance dans le domaine.

Il est nécessaire pour les studios de développement de jeux vidéos de trouver le bon

équilibre entre exploitation et exploration. L'exploitation est le développement d'un jeu sur des mécanismes déjà existants où les règles sont déjà établies par un autre jeu ou un opus précédent. L'exploration est la découverte de nouvelles mécaniques de *gameplay* ou la création de nouveaux outils de développement de jeux (comme un moteur de jeu). Le but est d'offrir aux clients des articles de qualité et attractifs. Cet équilibre est précaire et il est difficile pour un studio de développement d'investir sur les deux domaines à la fois.

1.3 Les moteurs de jeux

Un moteur de jeu (*Game engine*) est, typiquement, une suite logicielle contenant un framework de mécaniques de jeu permettant d'accélérer le développement d'un jeu vidéo. Il peut inclure une ou plusieurs facettes du développement du jeu allant de la physique, aux graphismes, aux sons, aux calculs, à la gestion des périphériques d'entrée/sortie jusqu'à la gestion automatique de l'intelligence artificielle. Voici une liste de certains des moteurs de jeu les plus connus accompagnés des jeux qui en font usage :

— *Unreal Engine* [**?**] développé par Epic Games : Fortnite, Outlast 2, Dragon Ball Fighter Z, Days Gone.

— *Unity Engine* [**?**] développé par Unity Technologies : 7 Days to Die, Cuphead, Ori and the Blind Forest, Pokemon Go.

— *CryEngine* [**?**] développé par Crytek : Far Cry, Crysis 3, Deceit, Mavericks.

— *Frostbite* [**?**] développé par Dice (EA) : Battlefield V, Anthem, FIFA, Need for Speed.

Chaque moteur de jeu présente des avantages et des inconvénients en fonction du type de jeu que l'on souhaite développer. Certains moteurs sont axés sur un type de jeu ou une plateforme en particulier afin d'être plus efficaces. L'innovation

dans les moteurs de jeu est aussi essentielle au développement de nouveaux jeux vidéos. Par exemple, un moteur de jeu plus récent pourra intégrer des éléments plus récents comme des graphismes plus réalistes ou détaillés, ou des intelligences artificielles plus évoluées.

1.4 Les types de *Gameplay*

L'exploration peut également consister en la création d'un nouveau mécanisme de *gameplay*. Ce genre d'innovation est plus facilement repérable par les joueurs et plus marquante en ce qui a trait à l'expérience de jeu.

Voici une liste non exhaustive des principaux types de *gameplay* présents dans les jeux vidéos :

— MMORPG (*Massive Multiplayer Online Role-Playing Games*) : jeu massivement multijoueur en ligne mettant en scène un jeu de rôle avec différents objectifs à remplir : *leveling*, histoire principale vs. secondaire, développement social pour atteindre ces objectifs sous forme de guilde, etc. (ex. : World of WarCraft, Black Desert Online).

— *Survival* : le joueur doit survivre aux événements présents dans le jeu. Il peut devoir subvenir à des besoins vitaux, construire de nouveau objets, ou survivre aux autres joueurs présents (ex. : Rust, Ark).

— Plateformes : un joueur contrôle un personnage qui se déplace dans un environnement de plateformes et doit avancer tout au long du niveau pour le compléter (ex. : Mario, Donkey Kong).

— Simulation de vie : le joueur simule un environnement de vie plus ou moins réaliste en fonction des objectifs du jeu. La simulation peut s'appliquer à un personnage ou à une ville entière. (ex. : Les Sims, SimCity).

— FPS (*First Person Shooter*) : le joueur, seul ou en équipe, doit battre des

ennemis (IA ou autres joueurs) à l'aide d'armes et d'équipements de combat (ex. : Call of Duty, Halo).

— *Beat-em up* : le joueur fait face à des vagues d'ennemis toujours plus fortes (ex. : Bayonnetta, God of War).

— RTS (*Real Time Strategy*) : des joueurs se font face dans un jeu où la gestion d'économie, de troupes et de population est omniprésente afin de battre les autres (Age of Empire, Starcraft).

— 4X (*eXplore, eXpand, eXploit, and eXterminate*) : proche du RTS, ce type de *gameplay* est fondé sur une gestion pointue de ressources et de populations afin de pouvoir battre les autres joueurs sous différents aspects et avec différents objectifs de victoire (population maximale, évolution de la société, critères financiers, etc.) (ex. : Civilization, Stellaris).

— MOBA (*Multiplayer Online Battle Arena*) : c'est un type de *gameplay* où le jeu d'action rencontre le RTS. Plusieurs équipes de joueurs sont téléportées sur un territoire, chaque joueur contrôle un personnage et les joueurs doivent détruire la base de l'équipe adverse (ex. : League of Legends, DOTA).

— *Battle Royal* : plusieurs dizaines de joueurs sont parachutés sur un territoire où ils trouvent des armes et doivent s'entretuer ; le dernier survivant est déclaré vainqueur (ex. : Fortnite, PUBG).

Les types de *gameplay* évoluent beaucoup. De nouveaux *gameplay*s apparaissent grâce aux recherches exploratoires. Certains modes de jeux à succès deviennent des catégories à part entière. Il est possible de combiner plusieurs modes de jeu afin de créer une nouvelle expérience. Ces divers types de *gameplay* sont classifiés en fonction du type de monde, des objectifs de jeu, des actions nécessaires, etc. Haitz et Law [**?**] ont mis en place une cartographie des genres afin de classifier les différents types de jeux en se référant à des caractéristiques précises des jeux et de leur *gameplay*. Cependant, il est difficile d'arriver à classifier tous les jeux

tellement les genres sont nombreux et entrecoupés. C'est pour cela que la plupart des jeux sont classifiés dans des catégories larges et sont ensuite différenciés par leurs diverses caractéristiques.

1.5 Objectifs de notre travail de recherche

[[(Guy T.) On reverra cette section ultérieurement. Une partie de cela devrait être dans l'introduction.]]

De nombreux défis entourent le développement de jeux vidéos et les outils et méthodes pour y parvenir sont déjà nombreux. Cependant, il devient de plus en plus important de chercher des méthodes afin d'accélérer le développement, sans pour autant impacter la qualité du contenu final. Un *Game designer* se doit d'être précis et concis dans ses descriptions et ses communications afin de communiquer un maximum d'informations et de pouvoir dédier plus de temps à l'exploration, à la recherche artistique et au développement d'un prototype.

Dans ce mémoire, nous allons présenter *Game Genesis* (GG), un profil UML [**?**] qui permet d'outiller le processus de pré-conception d'un jeu vidéo, facilitant ainsi la documentation et la rédaction des documents descriptifs du jeu vidéo en cours d'élaboration.

Destiné à être utilisé dans les premières phases du développement, *Breaktrough* et Pré-conception, ce profil UML permet l'accélération de la documentation à travers un outil visuel représentant les mécaniques et les objets du jeu. Ceci permet de produire un prototype jouable plus rapidement.

L'utilisation d'un profil UML permet d'apporter une structure précise et un cadre de travail rigoureux lors de la conception afin d'assurer la cohérence des modèles durant toute leur durée de vie.

Lors de sa conception, un jeu vidéo est destiné à évoluer rapidement et sa documentation se verra modifiée de nombreuses fois afin de toujours être la plus pertinente possible. Le standard UML — *Unified Modeling Language* (UML) — permet d'apporter un support visuel à cette documentation et permet également de rendre les modifications plus faciles en accédant aux objets concernés rapidement et efficacement.

Effectuer des *mind mapping* est un moyen simple et efficace permettant d'illustrer des idées de manière visuelle. Cependant, le manque de rigueur (modifications erratiques, contenu incohérents, manque de structure) et les supports utilisés (feuille de papier ou tableau) lors d'un travail de *mind mapping* n'assurent pas du tout la pérennité des informations stockées. Il est en effet compliqué de modifier une photographie d'un *mind map* lors d'une réunion s'étant tenue plusieurs semaines auparavant. De par sa structure imposée et précise le standard UML permet de structurer les données, de les stocker sous un format précis et de les maintenir facilement en communiquant sur les modifications effectuées. Ceci permet un stockage beaucoup plus pérenne des informations contenues dans le modèle.

UML étant un langage de modélisation largement utilisé et approuvé dans le domaine informatique, il est possible d'utiliser de nombreux outils déjà existants, d'assurer son intégration et sa réutilisation dans différents cadres et sous différentes formes.

Un modèle UML pourra également servir de support à la rédaction d'un — *Game Design Document* (GDD) — en organisant les mécaniques de jeu sous forme de catégories réutilisables. Le GDD est la «bible du design »[**?**] du jeu. Il permet de définir tout le contenu du jeu et toutes les informations nécessaires à son développement sur toute sa durée de vie.

UML étant reconnu comme un langage de modélisation de logiciels, il peut éga-

lement être possible d'utiliser les modèles créés en pré-conception afin de faciliter le développement informatique du jeu vidéo. Le modèle peut permettre de créer une structure de projet ou du pseudo code pouvant servir de squelette en début de développement.

CHAPITRE II

LES DOCUMENTS DE *GAME DESIGN*

Dans ce chapitre, nous allons présenter plusieurs documents de *Game design* qui sont présents dans un processus de développement de jeu vidéo. Ces documents peuvent être rédigés l'un à la suite de l'autre, ou indépendamment l'un de l'autre, et ils ne sont pas obligatoires dans chaque projet de jeu vidéo.

Les documents de *Game design* sont nombreux et leurs formes ainsi que leurs contenus sont souvent remis en question dans la littérature. Selon le niveau d'avancement du développement, certains sont plus utiles que d'autres mais la plupart des acteurs du domaine sont d'accord pour avancer qu'il n'existe pas de gabarit fixe pour un document de *Game design* [**?**]. En fait, un gabarit trop spécifique et précis serait une erreur à ne pas commettre, car il pourrait entraîner des difficultés de rédaction et d'adaptation face aux divers types de jeux que l'on peut rencontrer. Cependant, certains auteurs présentent des lignes directrices, un peu comme une recette de cuisine [**?**], qui peuvent être suivies afin que le GDD soit complet et respecte une certaines forme.

Dans sa présentation, Librande [**?**] montre des exemples de livrables de game design de différentes sortes. Il souligne que les documents de design lourds sont des sources d'informations importantes, réunissant tout le design dans un seul document et la création du document aide grandement à designer le jeu par la

suite. Cependant ces documents sont compliqués à maintenir, mettre à jours et à parcourir pour trouver une information précise.

Il présente également les «*design wikis*» qui apportent beaucoup de points positifs au design. Il est possible d'y avoir accès à n'importe quel endroit tant qu'une connexion est disponible. Les mises à jours sont simplifiées car la recherche l'est également : il devient alors possible de modifier des éléments directement au cours d'une réunion. Un *wiki* permet aussi la contribution de tous les membres de l'équipe de développement aux différents éléments car la modification est possible par de nombreux utilisateurs. Il est possible d'éditer les éléments par article et donc de ne pas avoir à prendre connaissance d'éléments non reliés au travail actuellement effectué. Il est facile de maintenir un historique des versions et des modifications pour justifier les modifications et les garder en mémoire. Cependant, un *wiki* requiert une maintenance constante et donc, souvent, une personne dédiée à sa maintenance. Les relations entre les éléments ne sont pas mises en avant et il est compliqué de mettre ensemble différentes sortes de représentation sous la forme de textes/images. Autres désavantages : les images doivent être traitées à l'extérieur du *wiki* avant d'y être réintégrées, et l'impression d'un article *wiki* n'est pas toujours adaptée à une lecture rapide des éléments.

2.1 Le concept document

2.1.1 : Le «*one-page*» de Librande [**?**]

[[(Guy T.) «Le concept document» ou «La documentation des concepts du jeu» ?]] [[(Ellen) C'est bien le concept document, le one page est un document concis décrivant le concept général d'un jeu, on n'entre pas dans les détails on ne passe que sur les lignes directrices]]

Librande [**?**] estime que plus un document est long, moins les utilisateurs s'y réfèrent. Afin de répondre à cette problématique il développe le *One-Page Design*. Un document *One page design* est une vue d'ensemble du jeu. Il est destiné à l'équipe de développement et aux acteurs décisionnels des studios. Il doit donc contenir les informations essentielles pour présenter le projet de jeu sous forme visuelle sur une seule page [**?**].

[[(Guy T.) Pas clair que c'est une autre proposition — Librande vs. Rogers ! ? A clarifier/expliciter !]]

Voici les éléments essentiels d'un *One-page design* tel que décrit par Librande [**?**] :

- Un titre représentatif du design
- Les dates des divers éléments, et ce afin de garder un historique
- Des espaces entre les informations, pour ne pas créer des blocs d'informations
- Une illustration centrale pour focaliser l'attention
- Sous l'illustration, possiblement une description et des textes explicatifs
- Pour des détails supplémentaires, des légendes autour de l'illustration — ces légendes peuvent être des illustrations, elles-mêmes accompagnées de notes
- Des barres latérales pour ajouter des *checklists*, des objectifs principaux ou des informations diverses

2.1.2 Le «*one-sheet*» de Rogers [**?**]

Dans son article Rogers présente le «*one-sheet*» comme vue d'ensemble du jeu. Il insiste sur le fait que ce document doit être intéressant, informatif et court. Voici les éléments que doit contenir le *One-Sheet* de Rogers [**?**].

- Le titre du jeu
- Les systèmes de jeu prévus

— L'age des joueurs visés

— La notation ESRB — voir plus bas

— Un résumé de l'histoire du jeu en se concentrant sur le *gameplay*

— Les modes du *gameplay*

— Les USP — voir plus bas

— Les jeux compétiteurs

[[(Guy T.) Jeux et logiciels ? Ou juste jeux ?]]

Le ESRB — *Entertainment Software Rating Board* (ESRB) — est un organisme d'autorégulation qui est à l'origine d'un système de notation et de règles de respect de la vie privée des jeux et logiciels. Son système de notation permet de définir à partir de points précis l'âge conseillé pour l'utilisation d'un jeu ou logiciel. C'est une notation que l'on retrouve systématiquement dans le domaine du jeu vidéo définissant le type de contenu présent dans le jeu et le public pour lequel est recommandé ce contenu : eC (*Early Childhood*), E (*Everyone*), E10 (*Everyone* 10^+), T (*Teen*), M (*Mature* 17^+), AO (*Adults Only* 18^+).

[[(Guy T.) Est-ce vraiment des «points» au sens de «pointage» (numériques) ? Ou juste des «arguments» — autre sens du terme «point» en anglais : selling point = argument de vente ! ?]]

Les USP — *Unique Selling Points* (USP) — sont des arguments de marketing représentant le jeu. Ils se situent à l'arrière de la boite de jeu ou sur le descriptif du jeu dans le cas d'une vente dématérialisée. Ce sont des points précis et courts attirant la curiosité de l'acheteur sans être trop descriptifs.

Les jeux compétiteurs liste les concurrents actuels du jeu, c'est-à-dire, les jeux déjà présents sur le marché. Cette liste doit contenir des jeux connus ou connaissant un grand succès, afin qu'ils soient représentatifs du type du jeu décrit par le *one-sheet*. Cela permet donc de donner une bonne idée de ce que le jeu sera. Présenter

une liste de jeux obscurs ou qui n'ont connus que peu de succès découragera les éditeurs qui liront le *one-sheet*.

[[(Guy T.) Ci-bas : Comme c'est une liste d'élements, il faut que tous soient des substantifs — et non des actions (verbes).]]

Selon Librande, la taille des éléments est importante dans un *One-page design* : les tailles indiquent l'importance de l'information. Si nécessaire, un *One-Page design* peut être étendu à la taille nécessaire pour contenir toute l'information, y compris sous forme d'un poster. Cependant, l'augmentation de taille de la page doit quand même permettre d'assurer la lisibilité — augmenter en taille ne doit pas conduire à intégrer trop d'informations, ce qui irait à l'encontre de la lisibilité.

2.2 Étendre le *One-page* vers un résumé

Une fois les premières étapes de *Game design* effectuées, le document *One-page* va être étendu afin de produire un document plus important. Ce second document va préciser les concepts du jeu vidéo en cours de création et va permettre de communiquer un niveau de détails plus précis.

Pedersen propose le *five pager* [**?**]. C'est un résumé du concept du jeu vidéo et une description du jeu à venir. Il comprend toutes les informations essentielles au cycle de vie du jeu sous ses aspects majeurs, tels que le *gameplay*, l'audience cible, le scénario, et les *features*. Le *five-pager* permet de présenter le jeu sous sa forme la plus basique à un décideur ou à un éditeur.

[[(Guy T.) Ci-haut : cela me semble contradictoire/ambigue/mélangeant de dire que le document de 5 pages présente le jeu sous sa forme «la plus basique», alors qu'on a au prélable produit un document d'une seule page. Reformuler ?]]

[[(Guy T.) Il faut éviter de mettre dans le corps du texte des longues séries de détails tel que le

contenu du *Ten-pager*. Dans le cas présent, je crois qu'une table serait plus appropriée.]]

[[(Guy T.) Pour une telle liste/table, préférable de ne pas mettre d'article, donc «Titre du jeu» plutôt que «Le titre du jeu». De plus, comme évoqué précédemment, il faut être uniforme, i.e., substantifs partout — et non pas mélanger substantifs et verbes (actions).]]

Quant à Rogers [**?**], il propose un document plus *fourni* avec son *Ten-pager*. Ce document comporte deux parties, qui se différencient radicalement en fonction du destinataire. Le contenu du document doit non seulement contenir des informations pour l'équipe de développement du jeu, mais également les informations nécessaires pour intéresser les éditeurs impliqués dans le projet. Rogers présente le contenu du *Ten-pager* sous la forme suivante indiquée dans le tableau **??**. Alors que les pages X à Y sont destinées plus spécifiquement à Z, les pages U à V sont destinées à W.

[[(Guy T.) Paragraphe ci-haut : Tu parles de deux parties avec destinataires distincts. Ce serait bien d'expliciter]]

2.3 Le *Game Design Document*

2.3.1 Le GDD, un moyen de communication

Le GDD est un des documents les plus complet trouvable dans le cadre du développement d'un jeu vidéo. Dans la littérature il est souvent qualifié de «Bible du design» [**?**] — bien que certains auteurs ne soient pas d'accord sur ce sujet [**?**].

Le GDD doit contenir, dans l'idéal, toute la vision du *game designer* sans laisser de doute sur ce que celui-ci veut représenter et souhaite voir dans le jeu. Peu importe le corps de métier de la personne lisant le GDD, celle-ci doit pouvoir se représenter au maximum le rendu final attendu. Le GDD doit donc pouvoir servir

- Page 1 : Informations générales
  - Titre du jeu
  - Systèmes de jeu prévus
  - Âge des joueurs visés
  - Notation ESRB
  - Calendrier prévisionnel de sortie
- Page 2 : Histoire
  - Résumé de l'histoire du jeu permettant de poser les premiers jalons de manière succinte et générale
  - Résumé du déroulement du jeu permettant de situer les actions du joueur dans l'histoire, les défis rencontrés par celui-ci, comment se déroule la progression, la place du *gameplay* dans l'histoire, les conditions de victoire, etc.
- Page 3 : Détails du personnage contrôle par le joueur
  - Histoire du personnage — caractère, traits importants de son apparence
  - *Gameplay* particulier associé au personnage, ses mouvements, ses armes
  - Maquette des contrôles proposés au joueur, par ex., représentation des raccourcis claviers
- Page 4 : *Gameplay*
  - Modèle de séparation de l'histoire (niveaux, chapitres, monde ouvert)
  - Scénarios particuliers (cinématique active)
  - Mise en avant des USP
  - Diagrammes et illustrations apportant des précisions
- Page 5
  - Images et description du monde du jeu
  - Découpage des zones de jeu
  - Liens entre les zones

Tableau 2.1: Contenu du *Ten-pager* selon Rogers [**?**].

- Page 6 : Expérience de jeu
  - Description des émotions et sensations que doit générer l'expérience de jeu
  - Description de l'interface de jeu et de la manière de les parcourir

    [[(Guy T.) A quoi réfère «les» dans «les parcourir» ? Pas clair !]]
- Page 7 : Mécaniques de *gameplay*
  - Mécaniques : Eléments avec lequel un joueur interagit pour effectuer des actions (par ex., levier pour ouvrir une porte)
  - Dangers : Eléments du monde pouvant tuer ou blesser le joueur mais sans IA (par ex., bloc de pierre qui tombe)
  - *Power-up* : Objets que le joueur récupère et utilise pour obtenir un avantage (par ex., champignon dans Mario)
  - Objets de collections : Objets que le joueur collecte mais qui n'influence par directement le *gameplay* (par ex., monnaie)
- Page 8 : Ennemis
  - Description des ennemis rencontrés par le joueur et qui sont contrôlés par une IA
  - Description des *boss* rencontrés
- Page 9 : Multijoueur et bonus
  - Description des succès collectionnables
  - Description des secrets découvrables
  - Description des interactions si le jeu est multijoueur
- Page 10 : Monétisation
  - Description du système de monétisation du jeu (Gratuit, Gratuit avec une boutique en jeu, payant à l'achat, etc.)
  - Description des boutiques et de leur contenu

Tableau 2.1: Contenu du *Ten-pager* selon Rogers [**?**] (suite).

de moyen communication entre les différentes équipes entourant le développement du jeu, et ce à toutes les étapes, depuis la création de l'idée jusqu'au post-mortem du projet.

2.3.2 Une structure souple mais un contenu précis

[[(Guy T.) Lorsqu'on indique un auteur, il suffit de mettre le nom de famille, sans le prénom.]]

Dans un article sur Gamasutra [**?**], Freeman établit une liste de 10 pratiques qu'il estime nécessaire de respecter pour décrire un GDD :

- Décrire le corps du jeu, mais également son âme
- Rendre le GDD lisible
- Etablir des priorité dans les tâches
- Rentrer au maximum dans les détails
- Démontrer et décrire les sujets
- Décrire le «quoi» mais également le «comment»
- Proposer des alternatives sur certaines réalisations [[(Guy T.) Pas clair : des alternatives de mises en oeuvres ?]]
- Donner une vie au GDD
- Préciser la vision du *Game designer* de façon suffisamment précise pour ne pas laisser d'éléments non décrits
- Mettre à disposition le GDD dans de bonnes conditions

2.3.3 Un GDD complet mais pas surchargé

[[(Guy T.) Ci-bas : j'ai reformulé la première phrase car c'était vague de parler «d'outils» et c'était bizarre de mélanger outils et liberté : A revoir !]]

Le GDD doit être un outil pour les *game designers*, tout en leur donnant la liberté de créer de nouveaux jeux, de nouveaux *gameplay*, de nouveaux concepts, donc sans les brider dans leur créativité. De nombreux travaux essaient de répondre au besoin de formalisation du GDD [**?** **?** **?** **?**] sans pour autant réussir à apporter une solution permettant d'englober tous les types de jeux, tous les types de métiers ou toutes les étapes de développement. Et de nombreux problèmes annexes se posent concernant l'universalité d'un modèle de GDD : Est-il complet ? Est-il assez précis ? Est-il assez souple ? Est-il utile à l'équipe ?

Prenons l'exemple d'un projet où le GDD est extrêmement complet, comportant tous les détails voulus par le *Game designer*. Le GDD peut alors devenir tellement complet qu'il en devient imposant, lourd à parcourir, difficile à maintenir et à modifier. Il perdra alors toute l'efficacité de design qu'il aurait pu donner, pouvant devenir un handicap de temps plutôt qu'une aide au développement [**?**].

Un GDD est l'atout majeur de la phase de production d'un jeu vidéo et pour qu'il soit utile durant toutes les phases celui-ci doit être impérativement complet et sans ambiguité [**?**]. Cependant il doit être assez souple afin de suivre le jeu dans son développement et suivre les changements nombreux et soudains qui peuvent arriver au cours du développement.

Dans son article [**?**], Freeman écrit que le GDD doit non seulement décrire le corps du jeu vidéo mais son «âme». Construire un GDD selon lui doit être un moyen de représenter le résultat attendu dans les moindres détails pour permettre aux équipes de travailler sur une idée précise. Faire appel à des outils graphiques plutôt qu'à des textes peut être un moyen d'apporter plus de précision à une idée. Mais le GDD doit être un moyen d'expliquer toutes les parties du jeu, le «quoi», mais également le «comment».

Cependant décrire l'intégralité des détails dans le GDD peut mener à des dérives

décrites par Rouse [**?**]. À vouloir apporter trop de détails dans un GDD, un *Game designer* peut perdre énormément de temps qui aurait pu être économisé avec plus de communication avec son équipe. Trop de détails peut également apporter trop de limitations aux corps de métier plus artistiques et brider leur créativité. Certaines parties du jeu peuvent également en obstruer d'autres : trop de détails dans la description d'une mécanique peut éclipser le fait qu'une autre n'est pas assez détaillée. Un GDD trop complet donne également l'impression que le projet est terminé et qu'il ne nécessite plus d'ajustements, cela nuit à l'évolution du jeu et aux modifications positives qui pourraient lui être appliquées.

[[(Guy T.) Il va falloir que je relise cette dernière section — en fait, le chapitre au complet — car j'ai l'impression qu'il y a certaines redondances/répétitions, mais aussi quelques aspects manquants dans le fil logique.]]

2.4 Conclusion

Dans ce chapitre, nous avons vu quatre sortes de documents de *Game design* : Les nommer ici, comme rappel/résumé/synthèse. Ces documents sont les plus répandus dans le monde du développement de jeux vidéos.

Selon le type de *gameplay*, la quantité d'informations ou la durée de vie du jeu, un GDD peut avoir une taille plus ou moins importante. La structure du GDD peut différer d'un *Game designer* à l'autre en fonction des habitudes de travail avec son équipe ou son expérience. Le contenu du GDD peut aussi être sous formes diverses : textes, graphiques, diagrammes, *mind mapps*.

C'est pour ces raisons qu'il est compliqué d'établir un gabarit précis pour la rédaction d'un GDD. La structure de tels documents n'est pas fixe et leur rédaction peut être effectuée en suivant des bonnes pratiques décrites par des *Game designer* expérimentés et trouvent réellement leur forme par la manière dont un *Game*

designer souhaite partager sa vision du futur jeu.

[[(Guy T.) Dernière phrase à relire!!!]]

CHAPITRE III

LE FRAMEWORK MECHANICS, DYNAMICS, AESTHETICS

La conception de jeu est le processus de réflexion nécessaire à la création d'un jeu vidéo ou de n'importe quelle sorte de jeu. C'est le chemin que suit le jeu depuis une idée jusqu'à sa réalisation. De nombreuses étapes permettent de mettre en place tous les aspects du jeu pour le mener à sa version jouable. La mise en place du monde dans lequel le joueur évoluera, les règles qu'il devra suivre pour avancer, les éléments graphiques qu'il rencontrera tout cela est défini lors de la conception.

Le jeu vidéo est cependant difficile à concevoir et surtout à documenter. Un logiciel classique répond au besoin d'un utilisateur qui s'en sert pour résoudre une problématique. Un jeu quant à lui doit apporter un but au joueur, l'envie de continuer à l'utiliser et de générer des émotions pour être apprécié. Il est très compliqué de définir une méthodologie précise pour décrire une émotion générée ainsi que tous les éléments techniques autour d'un jeu vidéo. C'est ce que le Framework *Mechanics Dynamics Aesthetics* (MDA) propose.

3.1 Le jeu : un échange entre *Game Designer* et Joueur

Un *Game Designer* créé un jeu dans le but de générer une expérience de jeu. Cependant l'utilisation que le joueur fera du produit fini est imprévisible [**?**]. Dans leur article Hunicke, LeBlanc et Zubek [**?**] découpent un jeu en trois aspects

distincts : les règles, le système et le «fun ». Ces trois aspects sont à l'origine des trois parties du Framework MDA comme indiqués dans la Figure **??**.

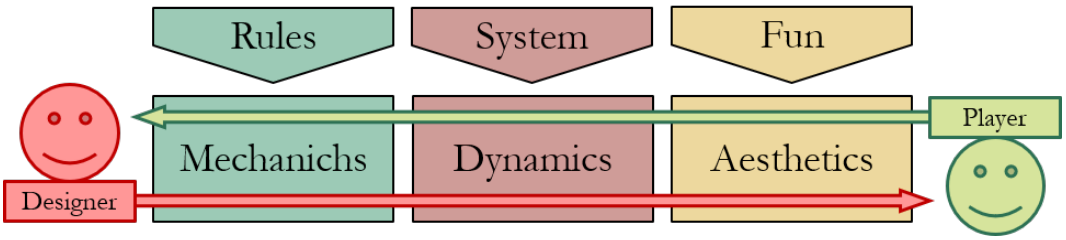

Figure 3.1: Contenu des parties du Framework MDA [**?**]

3.2 *Mechanics*

Les *Mechanics* du jeu sont tous les éléments composant le jeu au niveau de la représentation des données et des algorithmes. Les éléments sont dans des catégories qui permettent non seulement de les classer mais également d'aider à les définir plus précisément. Ces catégories peuvent comprendre des attributs et des spécificités qui sont appliquées aux éléments contenus dans les catégories.

Les *Mechanics* comprennent également les actions, comportements et mécanismes de contrôles mis à disposition du joueur. Cela peut correspondre aux mouvements d'un personnages, aux actions possibles sur des objets ou aux interactions entre les objets. Les règles sont aussi définies dans les mécaniques.

3.3 *Dynamics*

Les *Dynamics* sont les conséquences des *Mechanics*. Elles décrivent le comportement de l'exécution des *Mechanics* lorsqu'elles seront utilisées par le joueur [**?**]. Ces *Dynamics* sont importantes à prévoir car c'est elles qui permettront l'évolution du joueur dans le jeu.

Afin de définir les *Dynamics* il est possible d'analyser d'autres jeux (de même genre ou opus précédents) mais il est aussi possible d'effectuer des calculs statistiques sur des habitudes de jeu, étudier la psychologie des joueurs afin de prévoir leur *gameplay* en jouant sur les émotions.

3.4 *Aesthetics*

Les *Aesthetics* sont, selon Hunicke, LeBlanc et Zubek [**?**], «ce qui rend le jeu *fun* ». Ce sont toutes les émotions générées par le jeu et transmises au joueur via des mécaniques de jeu, les sons ou les graphismes. Ils classifient ces mécanismes en huit catégories :

— Sensation : le jeu comme plaisir des sens

— Fantaisie : le jeu comme imaginaire

— Narration : le jeu comme drama

— Challenge : le jeu comme un parcours d'obstacles

— Communauté : le jeu comme un réseau social

— Découverte : le jeu comme un territoire inexploré

— Expression : le jeu comme une découverte de soi-même

— Soumission : le jeu comme un passe temps

Le jeu final peut contenir une ou plusieurs catégories de «fun »et l'ensemble de l'expérience du joueur repose sur ces *Aesthetics* . Cette dernière partie du *Game design* est tellement importante qu'il est possible de concevoir et développer le jeu en ayant au préalable sélectionné un certain nombre de ces catégories et en les considérant comme objectifs à atteindre.

3.5 Les limitations et évolutions possibles

3.5.1 DPE : Le Framework Design, Play, Experience

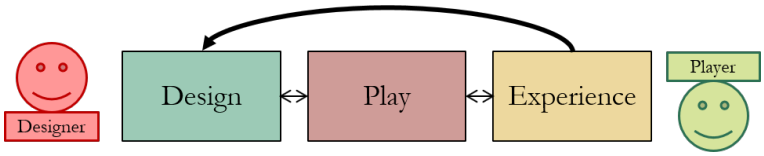

Figure 3.2: Contenu des parties du Framework DPE [**?**]

Le *Design Play Experience* (DPE) présenté dans la figure **??** est un framework reposant sur les mêmes principes que le MDA . Des modifications sont appliquées au MDA afin d'étendre ses capacités de design pour les jeux sérieux. Il étend le framework MDA afin d'y intégrer plus facilement les notions : d'apprentissage, de *story telling*, de *gameplay* et de composants technologiques spécifiques aux jeux sérieux. Dans la figure **??** sont présentés les différentes parties du DPE présentées par Winn [**?**].

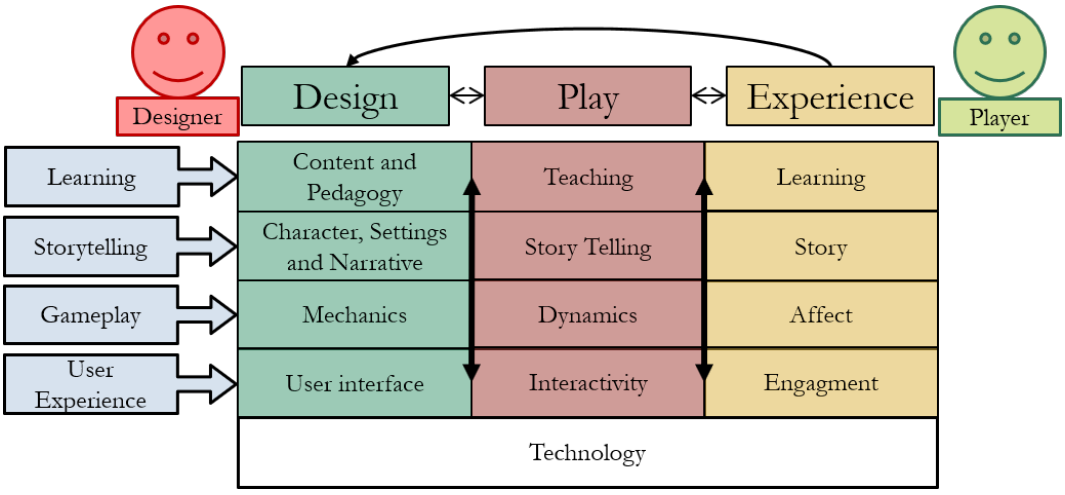

Figure 3.3: Framework DPE étendu [**?**]

Dans le DPE le *Game Designer* a le contrôle direct sur l'ensemble des catégories. Afin d'avoir le contrôle sur la partie Expérience le *Game Designer* définit des objectifs d'*Experience*. Dans la figure **??** c'est ce qui est représenté par la flèche entre Experience et Design. Cette flèche représente également le processus itératif du design décrit par Salen et Zimmermann [**?**]. La conception permet de produire un prototype, et l'expérience sur ce prototype permet de modifier le design afin que l'Expérience corresponde aux attentes du *game designer*.

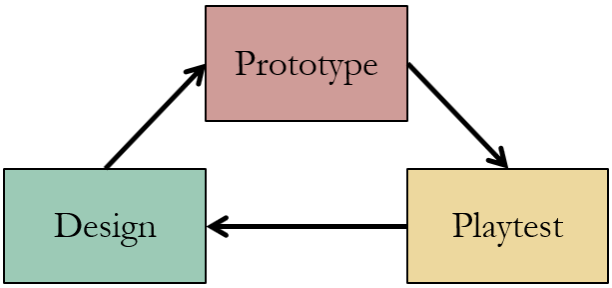

Figure 3.4: Processus de design itératif [**?**]

3.5.2 DDE : Le framework Design, Dynamics, Experience [**?**]

Le framework *Design Dynamics Experience* (DDE) décrit dans la figure **??** est présenté dans l'article de Walk *et al.* [**?**]. Ils apportent une vision critique sur le framework MDA et avancent que celui-ci néglige beaucoup d'aspects du design de jeux vidéos. Ils estiment également que le MDA se concentre trop sur les *Mechanics* et ne permet donc pas de décrire tous les types de *gameplay* présents sur le marché du jeu vidéo. Ce focus sur les *Mechanics* entraîne, selon Walk *et al*, un manque de contrôle du *game designer* sur les *Dynamics* et *Aesthetics* qui ne font que découler des *Mechanics*

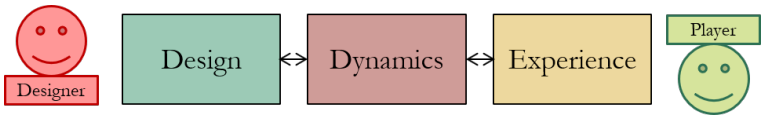

Figure 3.5: Contenu du Framework DDE [?]

Afin de définir le framework DDE ils effectuent un découpage différent et intègrent de nouvelles notions dans les catégories comme décrit dans la figure ??.

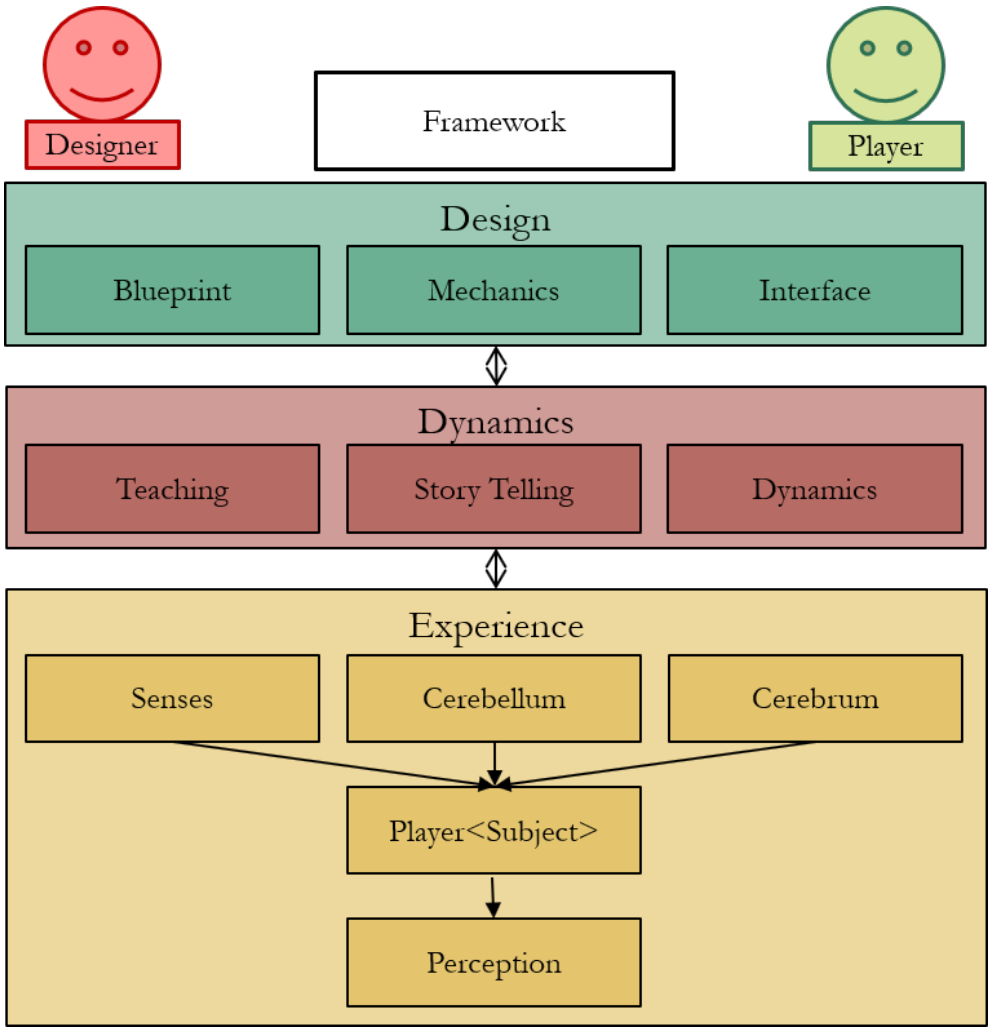

Figure 3.6: Framework DDE étendu [?]

Design

— *Blueprint* : Partie du design qui concerne les concepts du monde du jeu : culture, religion, physique, les différents sets de règles, les styles artistiques, le design narratif, le design de personnages et le design sonore qui ensemble créent l'expérience esthétique.

— *Mechanics* : Toute chose créant le jeu, plus précisément le code. L'architecture du code, la prise en charge des entrées/sorties, la prise en charge des objets, l'implémentation des règles de jeu et l'interaction entre les objets, et tous les éléments reliés au code. Cela comprend tous les éléments que le joueur ne perçoit pas dans son utilisation du jeu.

— Interface : Toutes les mécaniques qui ont pour but de communiquer le jeu au joueur. Les graphismes, le son, les réactions et interactions entre le joueur et le jeu, ainsi que celles bouclant sur le jeu lui-même. Cette partie comprend également les cinématiques les textes affichés et tout ce qu'il est possible de voir ou d'entendre dans le jeu.

Dynamics

La catégorie *Dynamics* du DDE correspond à celle présente dans le MDA, mais elle classifie les interactions de manière à les rendre plus précises. Cest ainsi que ces Dynamics se retrouvent en trois grandes catégories :

— Player <-> Game

— Player <-> Player

— Game <-> Game

Experience

Dans le MDA la troisième partie du framework est l'*Aesthetics* : tout ce que le joueur sent et ressent lors de son activité sur le jeu. La partie Expérience du jeu étend l'*Aesthetics* afin de prendre en considération que le joueur n'est pas une somme des émotions générées par le jeu. Le joueur devient un élément avec une expérience déjà présente avant l'utilisation du jeu ce qui peut modifier les émotions générées d'un joueur à l'autre. Une même couleur, un même son ou une même image peuvent générer différentes réactions de la part du joueur et la partie Expérience du DDE essaie de prendre en compte cela.

— Senses : expérience sensorielle du joueur du début à la fin du jeu.

— Cerebellum : les émotions ressenties par le joueur.

— Cerebrum : les challenges intellectuels et les décisions prises par le joueur.

— Player<Subject> : partie que le designer ne peut pas contrôler. Il peut se baser sur les trois premières catégories afin de prévoir une réponse spécifique du joueur. N'ayant aucun contrôle sur celle-ci il ne peut qu'estimer la réaction du joueur en fonction d'objectifs et de logiques psychologiques pour l'amener à la réaction souhaitée.

— Perception : ce que ressent réellement le joueur en fonction des trois premières catégories et de sa propre expérience et personnalité en tant que joueur. Cela comprendra son *gameplay*, le type de challenge qu'il perçoit, l'amusement qu'il ressent, la beauté qu'il perçoit, l'écho que génère l'histoire en lui, etc.

3.5.3 Le MDA est limité à la représentation des jeux vidéos

Dans son article Duarte [**?**] met en avant des difficultés que l'on rencontre à représenter des jeux qui ne soient pas des Jeux vidéos en faisant usage du MDA. Il

explique que les règles d'un jeu vidéo sont souvent implicites au type de *gameplay* ou de genre de classification de jeu vidéos. Cependant dans le cadre des jeux de plateau les règles ne peuvent pas être implicites et acquises par l'expérience. Elles doivent être explicites et explicables dans un manuel d'utilisation. Il fait l'analogie entre un *First Person Shooter* (FPS) et un jeu d'échecs. Dans un FPS un joueur sait à quoi s'attendre selon son expérience du genre. Il saura ou trouver les éléments et saura comment faire usage de l'interface graphique qui lui est présentée. Cependant un joueur se retrouvant devant un jeu d'échec pour la première fois ne pourra pas acquérir les connaissances requises pour jouer par l'expérience, les règles devront lui être expliquées à la base et l'expérience ne pourra lui apporter que des aspects tactiques du jeu.

3.6 Conclusion

CHAPITRE IV

LES PROFILES UML

Trouver une manière d'introduire le chapitre Ne modifient pas le metamodele Un profile étend le metamodel existant via des stereotypes, des tagged values et des contraintes. Pas possible de modifier les contraintes déjà existantes mais permet l'adaptation et la custromisation d'éléments existants du metamodel Les customisations sont définies dans un profile qui est ensuite appliqué à un package. Il est possible d'appliquer les profiles dynamiquement aux modeles et ils peuvent également être combinés pour appliquer plusieurs profiles simultanés au même modèle.

CHAPITRE V

UN PROFILE UML POUR AIDER À LA RÉDACTION D'UN GDD

Trouver une manière d'introduire le chapitre

CHAPITRE VI

EXEMPLE D'APPLICATION DU LANGAGE

Trouver une manière d'introduire le chapitre

6.1 PlayerUnknown's Battlegrounds (PUBG)[?]

PUBG est un jeu vidéo sortit en 2017 avec des millions de joueurs à travers le globe. Développé par PUBG Corporation (filiale de Bluehole, Inc. ou plus récemment de Krafton Game Union) en 2017 c'est un des premiers jeu standalone permettant à ses joueurs de participer à un "last man standing game" suivi de prêt par Fortnite (Epic Games). Ces deux jeux ont été les deux premiers standalone à populariser le Battle Royale et à le mettre à la portée de tous sans devoir développer ou installer des mods dans des jeux existants.

6.1.1 L'origine du Battle Royale

Le Battle Royale trouve ses racines dans le roman Battle Royale et son adaptation cinématographique. Un programme militaire de simulation de combat prend place da ns une République socialiste d'Extrême Orient complètement coupée du monde extérieur sans aucun droits civiques et extrêmement stable politiquement. Le programme consiste à un tirage au sort annuel de 50 classes de troisième qui

sont déplacées chacune vers une zone de combat. Au début de l'expérience tous les élèves sont réunis afin d'obtenir un briefing rapide de l'événement et se voient attribuer un sac contenant un objet (allant d'une arme à feu, à une arme blanche, à une fourchette ou une corde de luth). Ils sont ensuite livrés à eux-mêmes dans une zone donnée en ayant pour seule information : aucune règle n'est imposée et un seul survivant par groupe pourra être gagnant de l'expérience les autres devront être exterminés par n'importe quel moyen à disposition. Le champion obtient le droit de vivre aux frais de l'État pour le reste de ses jours et la reconnaissance comme Héros du pays.

6.1.2 La naissance de la popularité du Battle Royale

L'intérêt pour le principe du Battle Royale n'explose que plus tard avec le succès de la saga Hunger Games mettant en place l'histoire de Katniss Everdeen, une jeune adulte qui participe de son plein gré aux Hunger Games. Dans un État totalitaire séparé en castes regroupées dans des districts, deux enfants et/ou adolescents sont choisis au hasard dans chaque district afin de devenir les Tribus. Ils sont alors réunis à la capitale et lâchés dans une arène afin de participer à une téléréalité de match à mort diffusée partout dans le pays. Son succès a amené les développeurs du jeu Arma II à créer un mode de jeu basé sur les mêmes règles. Ce mode sera alors appelé DayZ et deviendra par la suite un jeu standalone. Un mode voit également le jour : Battle Royale. Développé par Brendan Greene pour Arma II et ensuite Arma III, ce mode gagne suffisament en popularité pour que celui-ci donne naissance au projet de jeu PlayerUnknown's Battlegrounds (PlayerUnknown étant le pseudonyme en ligne utilisé par Brendan Greene).

6.1.3 Les règles de PUBG

Un avion parcours une ligne droite sur une carte, les 100 joueurs de la partie doivent sauter de cet avion afin d'être parachutés à l'endroit de leur choix à condition qu'il soit à portée. Une fois arrivés au sol, les joueurs partent à la recherche d'armes et d'équipement dans des bâtiments et doivent s'entretuer jusqu'à ce qu'un seul joueur ne soit vivant et gagne ainsi la partie (il est également possible de jouer en duo ou en squad de 4 joueurs ou la dernière équipe debout devient gagnante). Afin de limiter la durée d'une partie, une zone circulaire est définie sur la carte une fois que tous les joueurs ont atterri et celle-ci se réduit par étapes au cours de la partie. Les joueurs en dehors de la zone subissent une certaine quantité de dégâts toutes les secondes s'ils restent en dehors du cercle et ces dégâts augmentent au fur et à mesure que le cercle se resserre.

6.1.4 Les Battle Royale dans le monde du jeu vidéo

Ces dernières années une quantité astronomique de Battle Royale voit le jour dans le jeu vidéo. De Fortnite (Epic Games) à PUBG, en passant par Apex Legends (Respawn Entertainement), Ring of Elysium (Tencent Games) pour ne citer que les plus importants. Des dizaines de jeux voient le jour ou se réinventent afin de coller à la mode des Battle Royale. Les plus grandes licences de FPS (First Personal Shooters) s'alignent également à cette mode, c'est ainsi que voient le jours les modes de jeu Z1BR (H1Z1), Firestorm (Battlefield V), Blackout (Call of Duty Black Ops IV). Mais cette offre répond à une demande phénoménale du public pour ce mode de jeu qui s'impose dans le marché à la même place que le type de jeu MOBA (Multiplayer Online Battle Arena) qui était grand favoris du public depuis une dizaine d'années à travers les jeux League Of Legends ou Dota 2. Avec le temps les Battle Royale tentent de se réinventer et de trouver de nouveaux

publics en gardant le principe de Battle Royale mais en apportant d'autres univers. C'est ainsi que naissent Last Tide et son univers aquatique immergé, Fall Guys : Ultimate Knockout et son univers colorés où de petits personnages à la physique étrange se battent pour une couronne, Cuisine Royale, qui était à l'origine une blague de développeurs mais qui a rencontré un succès, dans lequel les personnages se battent à l'aide d'ustensiles de cuisine.

6.2 Représentation d'une partie des mécaniques de PUBG

6.2.1 Les données

Données difficiles à obtenir de façon fiable. Informatiosn extraites de https ://pubg.gamepedia.com/ Wikipedia régulièrement mis a jours Extraction des informations via les fichiers de jeu minés

6.2.2 Le profile

Arbre des stereotypes découpé avec explications

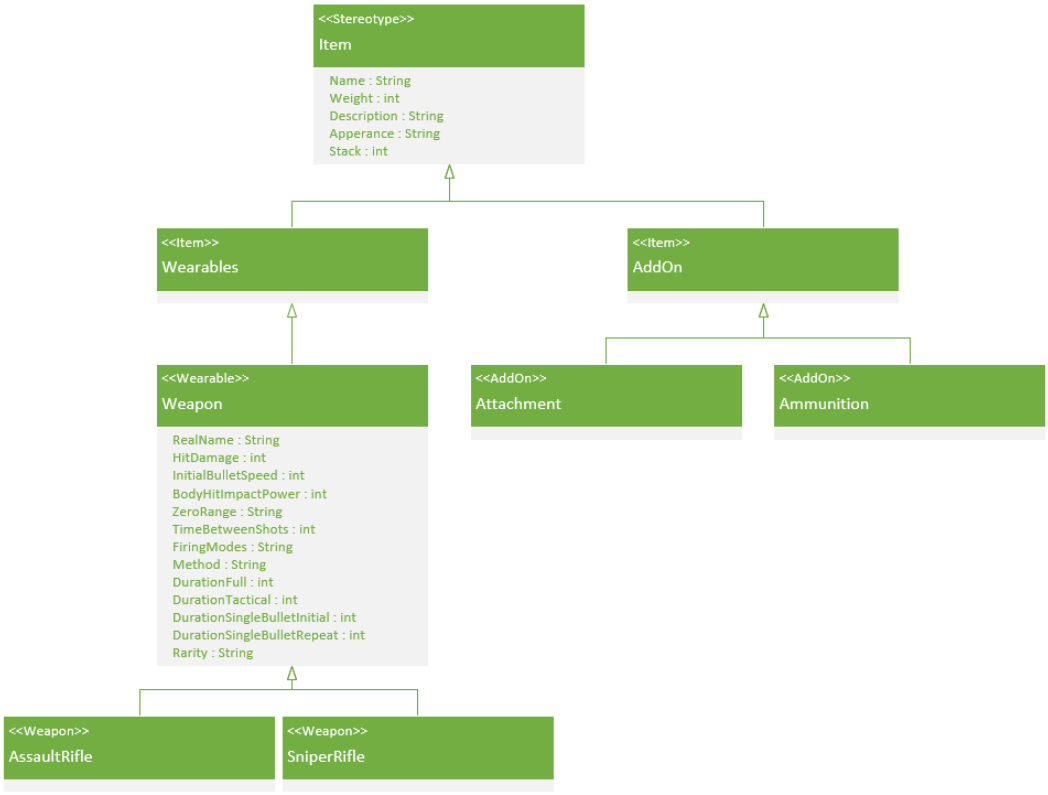

Figure 6.1: Racine des stéréotypes concernés par l'exemple

Focus sur la section concernée par la partie decrite plus bas (attachments)

6.2.3 La modélisation

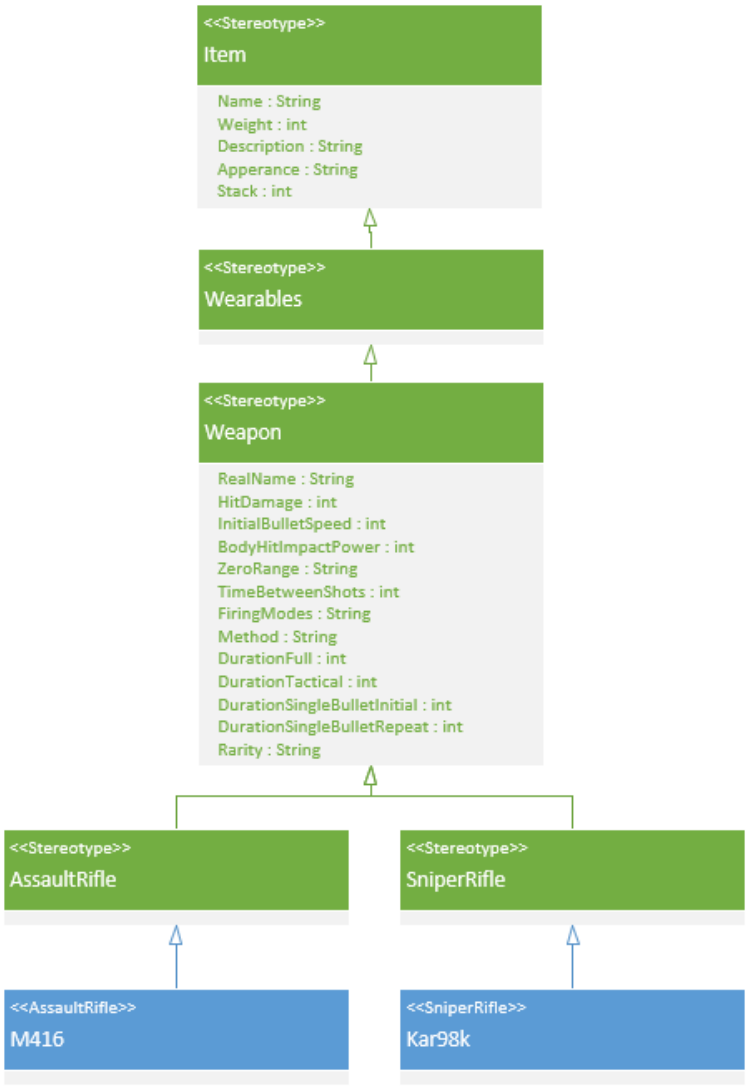

Figure 6.2: Classes des "Weapon" concernées par l'exemple

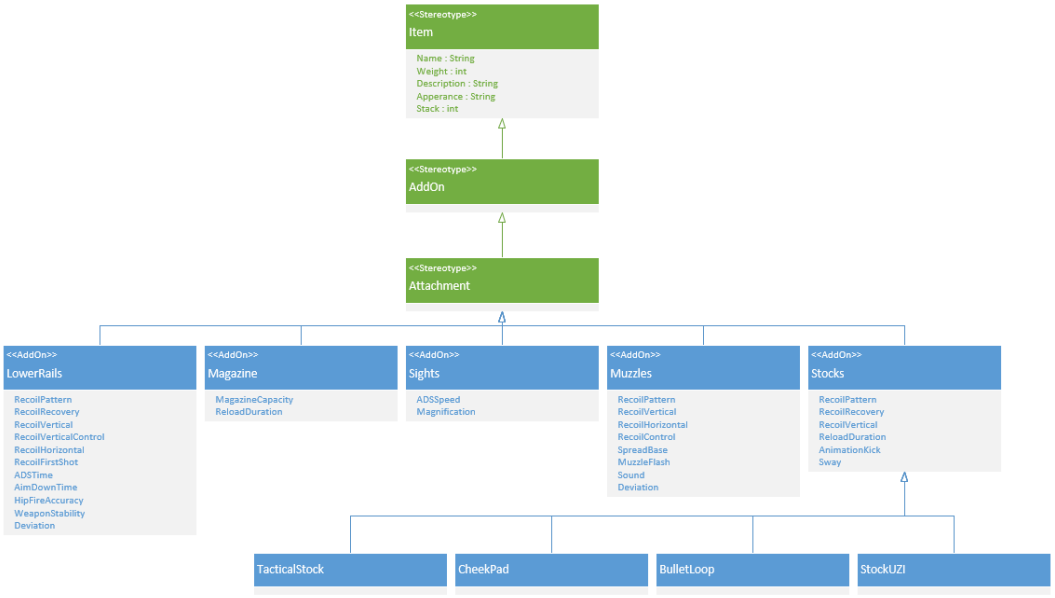

Figure 6.3: Arbre des "Attachment" concernés par l'exemple

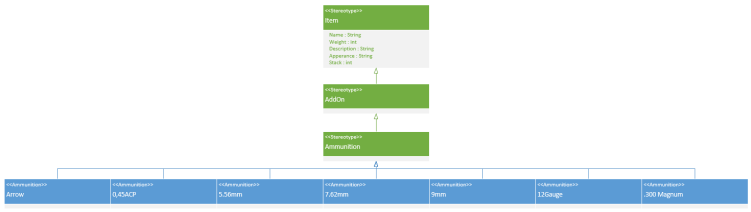

Figure 6.4: Arbre des "Ammunition" concernés par l'exemple

Afin de pouvoir visualiser les interactions entre les éléments il est nécessaire de nommer les associations entre les classes. Afin de mettre en place ces nommages d'associations un groupe de stéréotypes a étét mis en place.

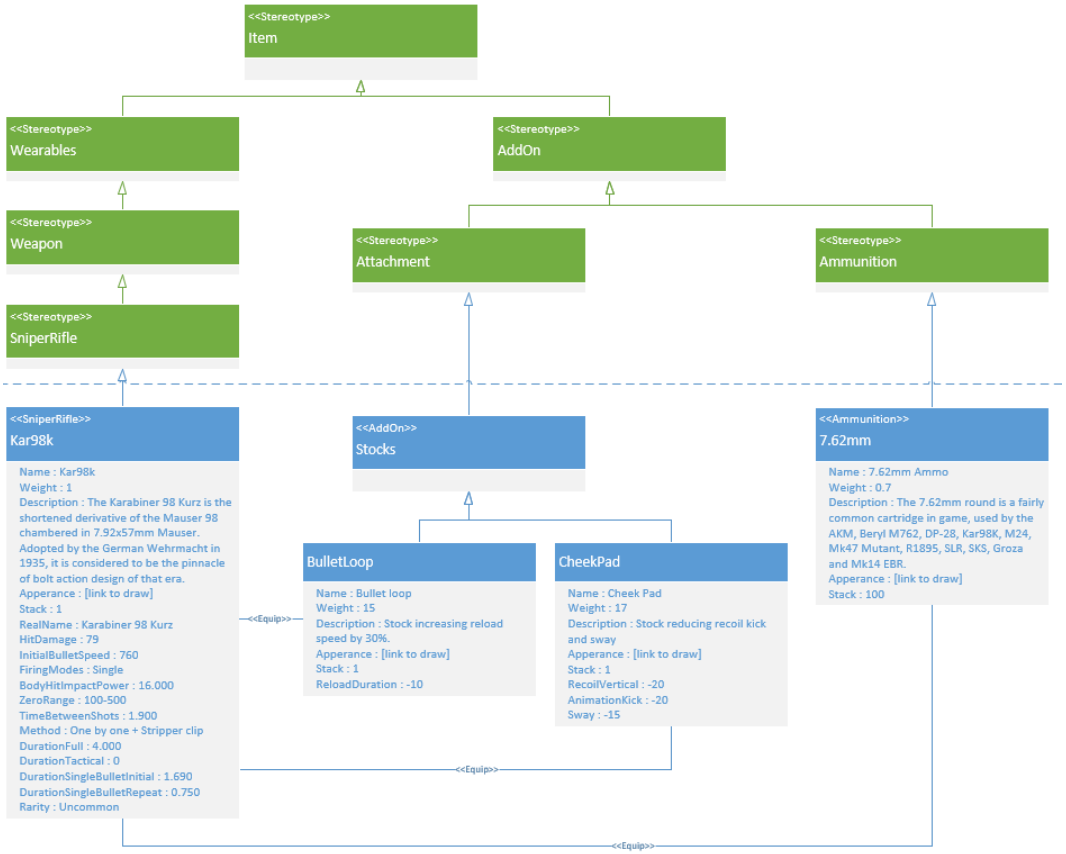

Figure 6.5: Arbre du Kar98k, avec ses Stocks et ses Ammunitions concernés par l'exemple

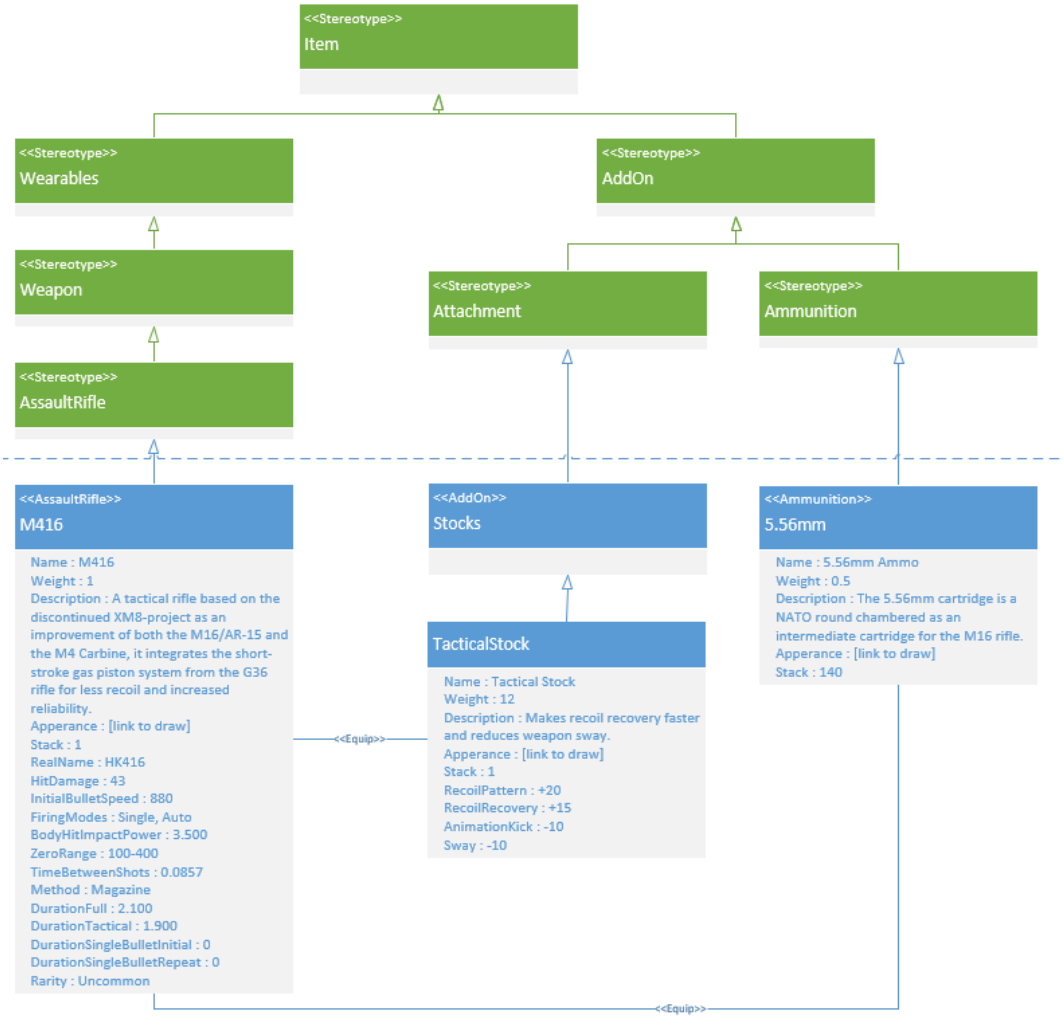

Figure 6.6: Arbre du M416, avec ses Stocks et ses Ammunitions concernés par l'exemple

Figure 6.7: Arbre des "Interactions" concernées par l'exemple

6.2.4 Synthèse sous forme de GDD

Rédaction du GDD avec les informations de la modélisation

6.3 Evolutions possibles

6.3.1 Outil de manipulation visuelle du profile UML

6.3.2 Transformation du modèle en GDD

6.3.3 Accélérateur de description des mécaniques

CONCLUSION

APPENDICE A

DIAGRAMMES DU PROFILE UML CRÉÉ

Item

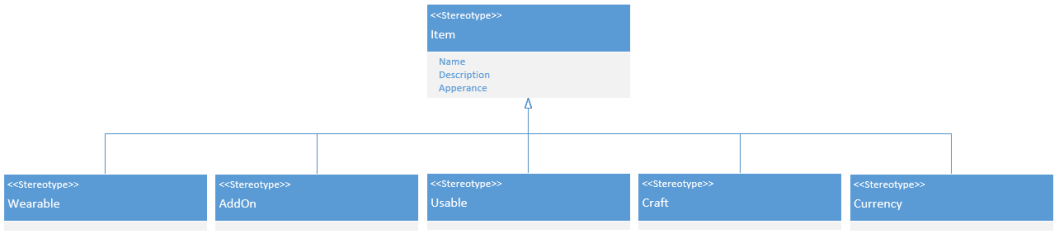

Figure A.1: TITRE

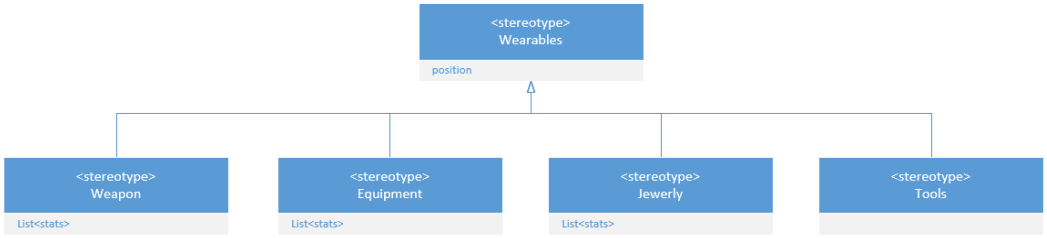

Figure A.2: TITRE

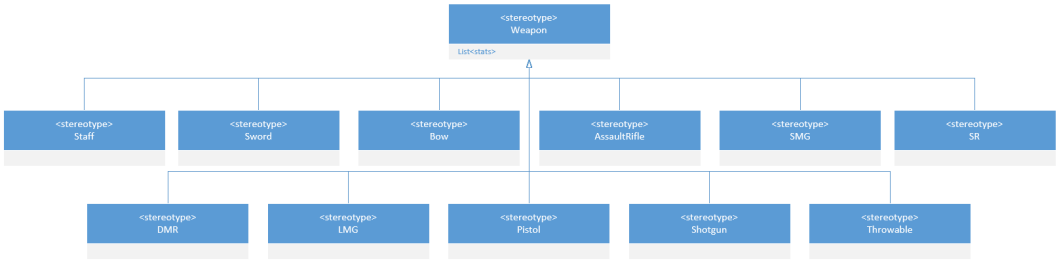

Figure A.3: TITRE

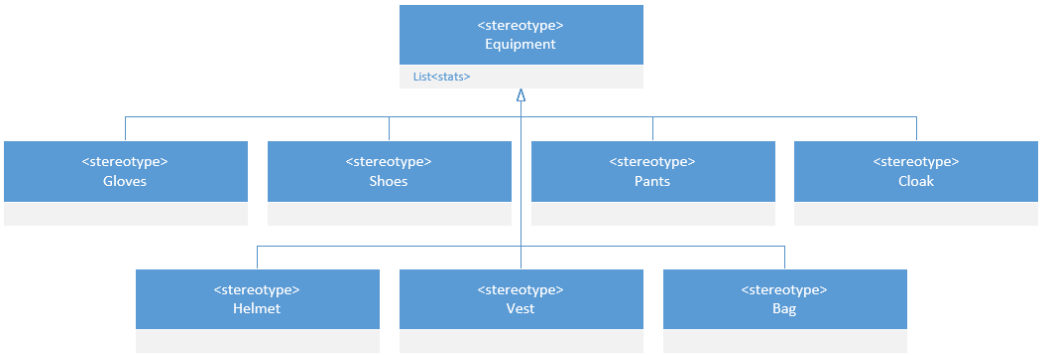

Figure A.4: TITRE

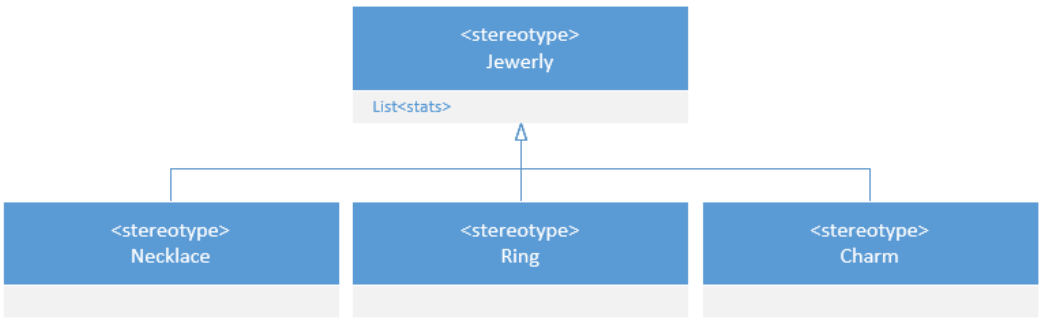

Figure A.5: TITRE

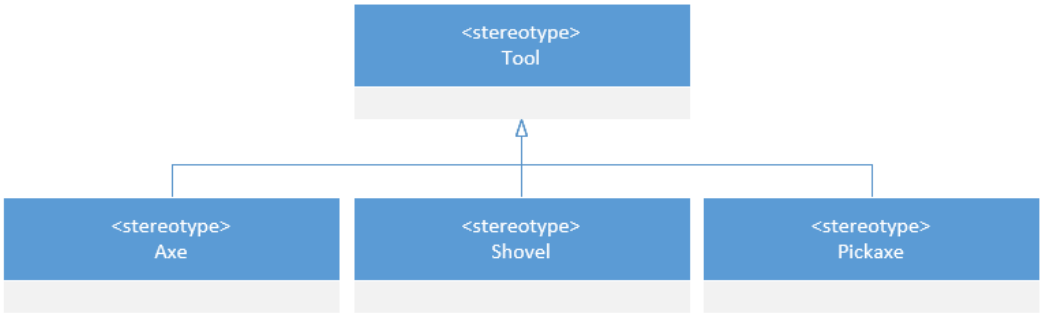

Figure A.6: TITRE

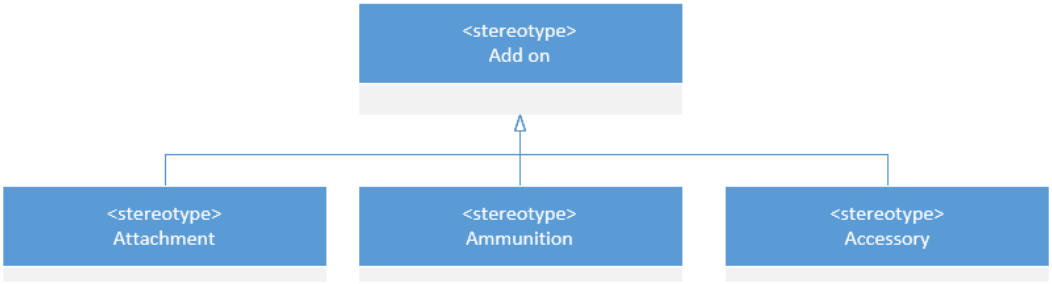

Figure A.7: TITRE

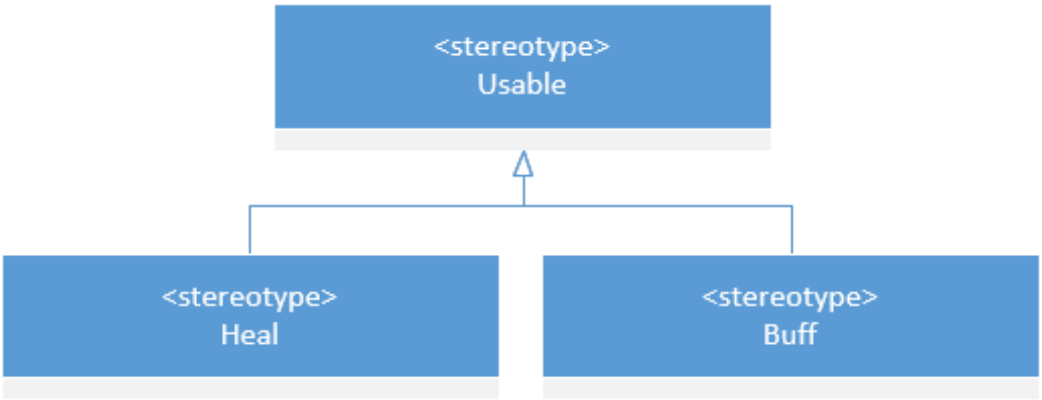

Figure A.8: TITRE

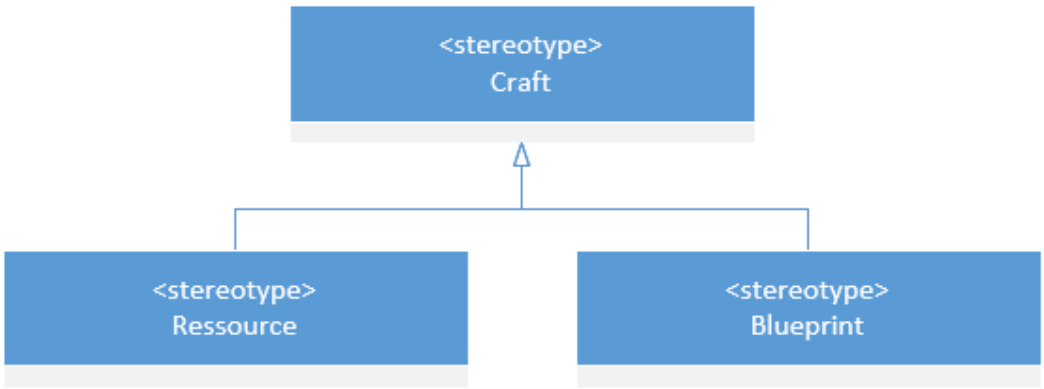

Figure A.9: TITRE

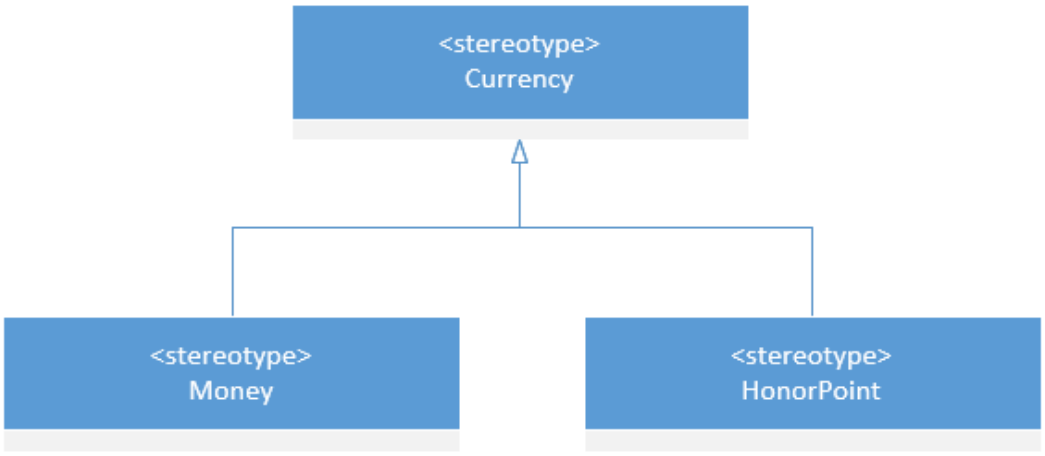

Figure A.10: TITRE

Animate

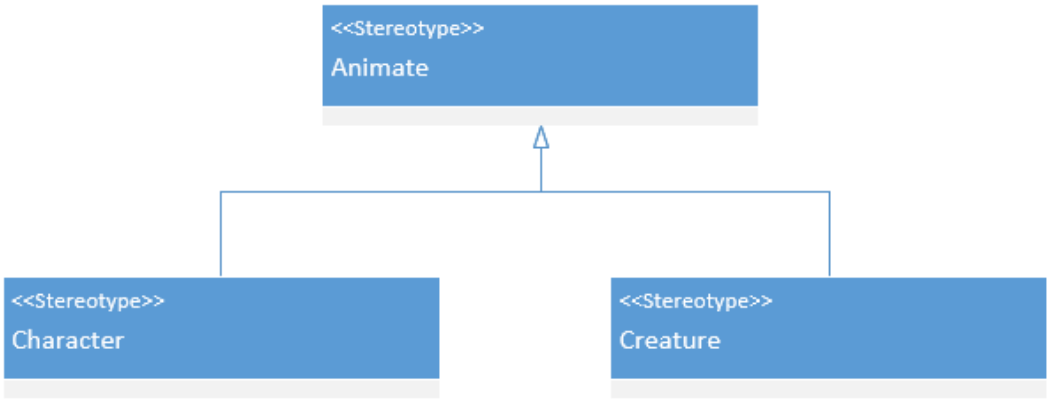

Figure A.11: TITRE

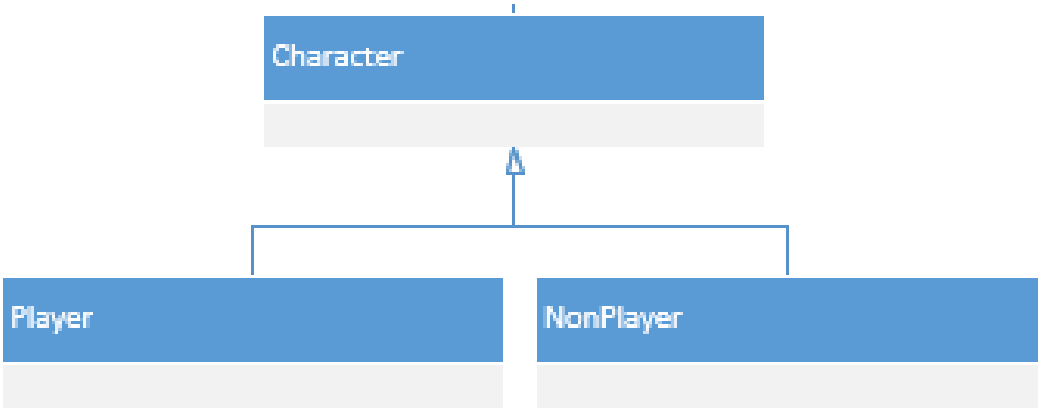

Figure A.12: TITRE

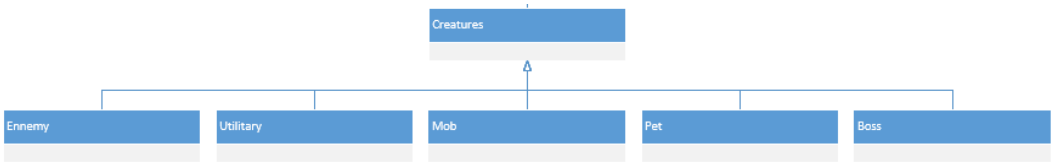

Figure A.13: TITRE

Character Sheet

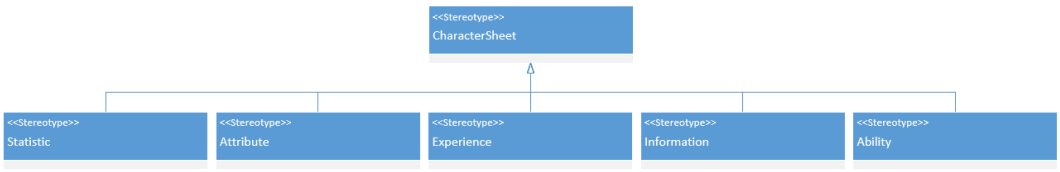

Figure A.14: TITRE

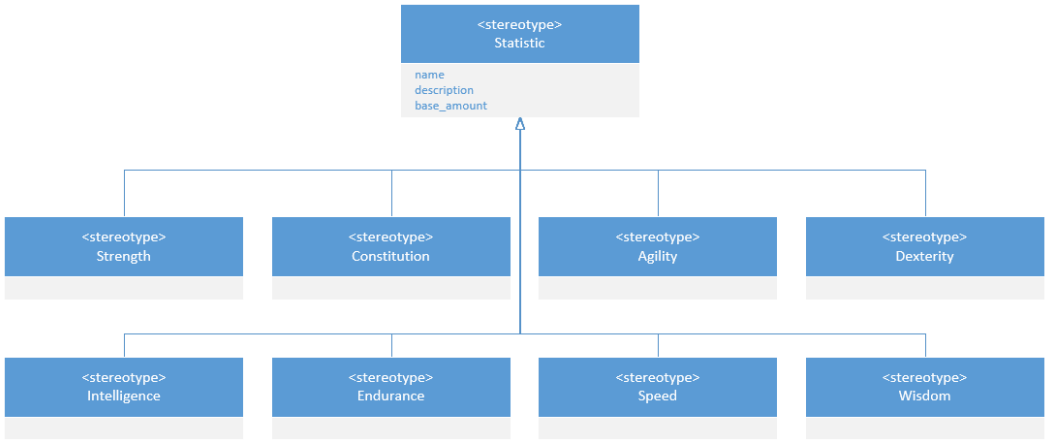

Figure A.15: TITRE

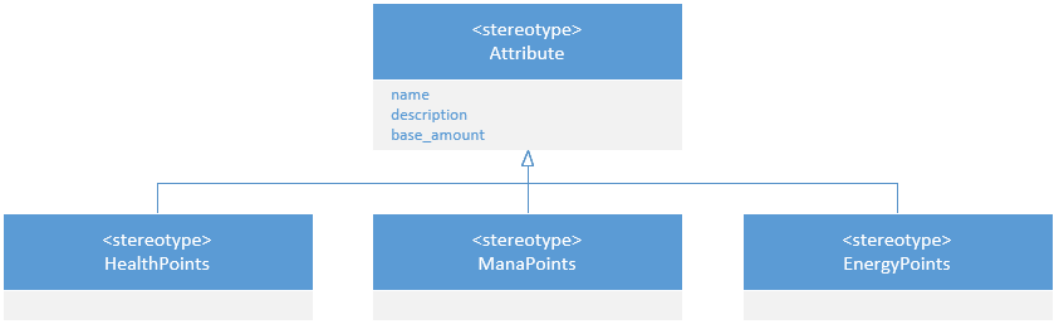

Figure A.16: TITRE

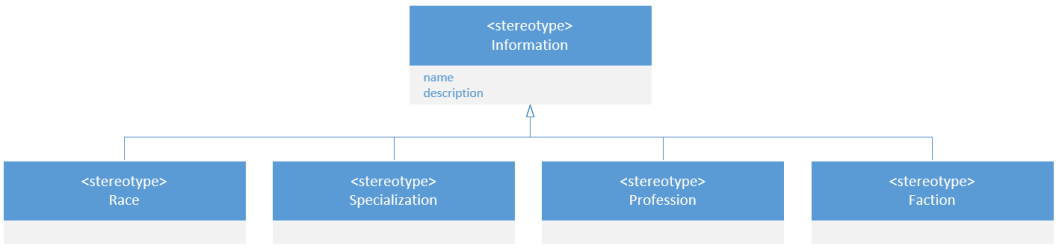

Figure A.17: TITRE

LORE

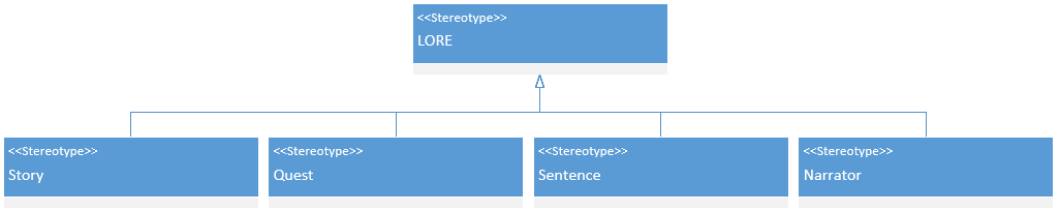

Figure A.18: TITRE

World

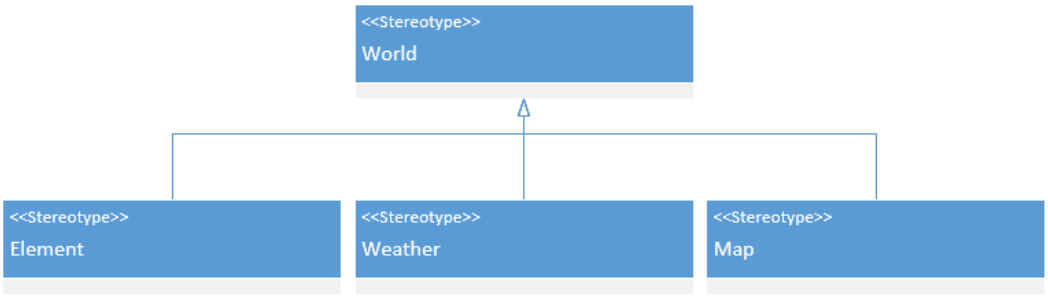

Figure A.19: TITRE

Interaction

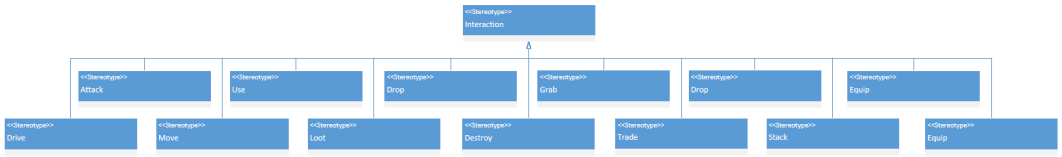

Figure A.20: TITRE